IBM InfoSphere Information Server
Version 11 Release 3

Connectivity Guide for Dynamic Relational Database Access

SC19-4271-00

IBM InfoSphere Information Server
Version 11 Release 3

Connectivity Guide for Dynamic Relational Database Access

SC19-4271-00

Note

Before using this information and the product that it supports, read the information in “Notices and trademarks” on page 79.

Contents

Chapter 1. Configuring access to data sources

To configure database connectivity, you must install database client libraries and include the path to these installed libraries in the library path environment variable. Apart from the library path, for certain database types you must set additional environment variables on the engine tier computer.

About this task

You must install the client libraries and include the directory that contains the database client libraries in the library path environment variables. For more information about setting environment variables, see "Setting the library path environment variable" on page 19.

Some 64-bit database client software installations include 32-bit version and 64-bit version of the client libraries. In this case, you must set the library path to point to the libraries that have the bit level that matches the bit level of the engine tier installation. Otherwise, when a job that uses the connector that requires the client libraries runs, an error is reported because the stage library is not able to load the database client libraries.

Procedure

1. Install database client libraries.
2. Configure access to data sources.

Configuring access to DB2 databases

To use the DB2 Connector stage in a job, you must configure DB2 environment variables and set the privileges for DB2 users. The DB2 connector connects to your databases by using the DB2 client on the InfoSphere® DataStage® nodes.

Before you begin

- Confirm that your system meets the system requirements for InfoSphere Information Server. Make sure that you are using a supported version of IBM® DB2®. For more information about system requirements, see http://www.ibm.com/software/data/infosphere/info-server/overview/.
- Install the IBM DB2 client on all InfoSphere DataStage nodes, and make sure that the client is working correctly.
- Use the DB2 Configuration Assistant to test the DB2 client and server connection. If the DB2 client fails to connect to the DB2 server, jobs that use the DB2 Connector stage also fail.
- Catalog each database in the DB2 client.
- InfoSphere DataStage runs many processes for each job. Ensure that DB2 resources, configuration parameters, and manager configuration parameters are configured properly.
- Make sure that the **DB2_PMAP_COMPATIBILITY** registry variable is set to `ON` to indicate that the distribution map size remains 4,096 (4-KB) entries. Though DB2® version 9.7 database for Linux, UNIX, and Windows supports distribution map entries up to 32,768 (32 KB), the DB2 connector supports only 4-KB entries in distribution maps.

- If you plan to use the DB2 connector with DB2 for z/OS in jobs with sparsely arriving data (such as jobs that use the Change Data Capture stage), ensure that the idle timeout value set in the DB2 **IDTHTION** subsystem parameter is longer than the longest expected interval of inactivity for the DB2 Connector stages in the job.

Procedure

1. Grant the InfoSphere DataStage users SELECT privileges on the following tables:

Table 1. Required SELECT privileges

DB2 product	Tables that require SELECT privileges
DB2 Database for Linux, UNIX, and Windows	SYSCAT.COLUMNS SYSCAT.KEYCOLUSE SYSIBM.SYSDBAUTH SYSCAT.TABLES
DB2 for z/OS	**Note:** Before the data is loaded to data to DB2 for z/OS, make sure the user has the GRANT ALL access on SYSIBM.SYSPRINT: SYSIBM.SYSCOLUMNS SYSIBM.SYSINDEXES SYSIBM.SYSKEYCOLUSE SYSIBM.SYSKEYS SYSIBM.SYSPRINT SYSIBM.SYSTABLESPACE SYSIBM.SYSTABLES SYSIBM.SYSTABLEPART SYSIBM.SYSUSERAUTH
DB2 Database for Linux, UNIX, and Windows and z/OS	SYSIBM.SYSDUMMY1 SYSIBM.SYSVIEWS

2. On DB2 for z/OS, ensure that the DBA runs the DSNTIJSG installation job to install the **DSNUTILS** stored procedure. The **DSNUTILS** stored procedure is required to start the bulk loader on DB2 for z/OS. For more information, see http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2z9.doc.inst/src/tpc/db2z_enabledb2supplstprocs.htm
3. Set the **DB2INSTANCE** environment variable to the instance in the DB2 client in which you cataloged the target database.

   You must set the **DB2INSTANCE** environment variable even if the stage accesses the default DB2 instance. The instance that is specified in the **DB2INSTANCE** environment variable becomes the default instance that is used by the connector. If you want to use a DB2 instance other than the default, then enter the name of that instance in the **Instance** property of the DB2 connector in the **Properties** tab. The DB2 client installs the default instances.

Table 2. Default instance installed by the DB2 client

Operating System	DB2 Instance
Linux or UNIX	`db2inst1`
Microsoft Windows	`DB2`

4. Add the path to the directory that contains the client libraries to the library path environment variable. The default path for client libraries is shown in the table.

Table 3. Default path for DB2 client libraries

Operating System	DB2 Instance
Linux or UNIX	`/opt/IBM/db2/V9/lib64`
Microsoft Windows	`C:\IBM\SQLLIB\BIN`

5. Optional: If the globalization map name for the DB2 connector job does not match the current system locale on the engine tier, then set the **DB2CODEPAGE** environment variable to a codepage corresponding to the map name. The DB2 code page can also be set by using a DB2 registry variable.

Configuring the DB2 native ODBC driver on AIX

To configure and use the DB2 ODBC driver on AIX operating systems, you must modify the DB2 environment variables and the **ODBCINI** environment variable.

Before you begin

- Confirm that your system meets the system requirements and that you are using a supported version of IBM DB2 database systems. For more information, see System Requirements.

Procedure

1. Update the 64-bit ODBC driver for DB2 on AIX operating systems so that the DataDirect driver manager can load the ODBC driver:
   a. Open the `db2o.o` driver file, which is in the *$DB2_HOME*`/sqllib/lib64` directory.
   b. In the `db2o.o` driver file, add a link to the `db2o.o.so` file. For example, you can add the following link: `ln -s db2o.o db2o.o.so`
2. Add the following entry to the `$ODBCINI` file. The DataDirect driver manager uses the `DriverUnicodeType=1` entry to work with the ODBC driver for DB2.

```
****************************************************
****************************************************
Driver=/home/db2inst1/sqllib/lib64/db2o.o.so
DriverUnicodeType=1
```

3. Set the **DB2INSTANCE** environment variable to the instance in the DB2 client in which you cataloged the target database.
4. Add the path to the directory that contains the client libraries to the library path environment variable.

Configuring access to Greenplum databases

You can configure access to a Greenplum database by configuring an ODBC data source definition (DSN) for Greenplum and by adding the directory location of the Greenplum parallel file distribution program (`gpfdist`) to the system path. The Greenplum Connector stage uses ODBC to connect and to execute statements and uses the `gpfdist` program to exchange data with the Greenplum database.

Configuring ODBC access to Greenplum databases on Linux and UNIX

To connect to a Greenplum database, you must first configure an ODBC data source definition (DSN) for the database by using the IBM Greenplum Wire Protocol ODBC driver.

Before you begin

- Ensure that the ODBC driver for Greenplum libraries is installed.
- Ensure that the path to the directory that contains the ODBC libraries are added to the library load path environment variable. On Linux and UNIX, the default path is `/opt/IBM/InformationServer/Server/branded_odbc/lib`. The following table lists the name of the library path variable for the different operating systems.

Table 4. Library path variables

Operating system	Library path variable
HP-UX	LD_LIBRARY_PATH or SHLIB_PATH
IBM® AIX®	LIBPATH
Linux	LD_LIBRARY_PATH

- Ensure that the *ODBCINI* environment variable is set to point to the `.odbc.ini` file, which contains the ODBC DSN definitions.

  Note: The *ODBCINI* environment variable is automatically set in the `dsenv` script as part of the InfoSphere Information Server installation process.

Procedure

1. Add a new ODBC DSN definition for the Greenplum Wire Protocol to the `.odbc.ini` file.
2. Specify the Hostname and PortNumber to the host and port where Greenplum database server is running.
3. Specify the Database name to default database to use for connections using the new ODBC DSN.
4. Save the `.odbc.ini` file.

Example

```
[Greenplum_DEV_SERVER]
Driver=/opt/IBM/InformationServer/Server/branded_odbc/lib/VMgplm00.so
Description=DataDirect 7.0 Greenplum Wire Protocol
...
Database=gp_dev
HostName=gp_host
PortNumber=5432
```

where, *gp_dev* is the name of the Greenplum database the DSN connects to and *gp_host* is the host name where the Greenplum server resides.

For information about configuring other driver options, see the Greenplum Wire Protocol Driver chapter in the *DataDirect Connect Series for ODBC User's Guide*.

Configuring ODBC access to Greenplum databases on Windows

To connect to a Greenplum database, you must first configure an ODBC data source definition (DSN) for the database by using the IBM Greenplum Wire Protocol ODBC driver.

Before you begin

- Ensure that the ODBC driver for Greenplum libraries is installed.

Procedure

1. Start the Microsoft ODBC Data Source Administrator.
   - On a 32-bit Windows computer, click **Start > Control panel > Administrative Tools > Data Sources (ODBC)**
   - On a 64-bit Windows computer, navigate to `C:\Windows\SysWOW64\odbcad32.exe`.

     Note: On Windows, InfoSphere® Information Server is a 32-bit application. Even on a 64-bit Windows computer, the connector is running as a 32-bit application. Therefore you must use the 32-bit version of the ODBC Data Source Administrator as the Greenplum connector will not be able to locate DSN definitions created in the 64-bit ODBC Data Source Administrator.
2. On the System DSN page, click **Add**.
3. On the Create New Data Source page, select the IBM Greenplum Wire Protocol driver and click **Finish**. For information about configuring the driver options, see the Greenplum Wire Protocol Driver chapter in the *DataDirect Connect Series for ODBC User's Guide*.

Greenplum parallel file distribution program (gpfdist)

The Greenplum Connector stage exchanges data with the Greenplum server by using the Greenplum file distribution program, which is called `gpfdist`.

The `gpfdist` program runs on the database client, and it must be installed on the InfoSphere Information Server engine tier computer. For data to be transferred by using the `gpfdist` protocol, a network route must be present to enable bidirectional access by using an IP address and optionally the presence of a DNS server to facilitate the name resolution. The connector invokes a `gpfdist` process on every physical computer node and creates the external table. The host of the external table data is identified by the `fastname` entry in the parallel engine configuration file (`$APT_CONFIG_FILE`). In order for the connector to invoke `gpfdist` on each engine tier, the location of `gpfdist(%GPHOME_LOADERS%\bin)` must be in the system path. In addition, the location of `gpfdist` dependent libraries (`%GPHOME_LOADERS%\lib`) must be in the system library path. On Windows, the system environment variable *PATH* is updated in the Advanced system settings. On Linux, the *PATH* environment variable is updated in the `dsenv` script.

Note: On Windows, the Greenplum installer adds `%GPHOME_LOADERS%\bin` and *%GPHOME_LOADERS%\lib* to the *PATH* system environment variable. Verify that these directories are in the *PATH*.

For manually adding `%GPHOME_LOADERS%\bin` and *%GPHOME_LOADERS%\lib* to the *PATH* system environment variable see the topic on setting the library path environment variable.

Configuring access to Informix databases

In jobs that contain the Informix® CLI, Informix Load, Informix XPS Load, or Informix Enterprise stages, you must set environment variables so that the jobs can access Informix databases.

Configuring access for the Informix CLI, Informix Load, and Informix XPS Load stages

For the Informix CLI, Informix Load, and Informix XPS Load stages to access Informix databases, you must set the values of environment variables and add data source name (DSN) entries to the `.odbc.ini` file.

Before you begin

- Install the IBM Informix server.

Procedure

1. Set the `INFORMIXDIR` environment variable to point to the installation directory of the IBM Informix server.
2. Make sure that the `PATH` environment variable contains `$INFORMIXDIR/bin`.
3. Make sure that the library path environment variable contains`$INFORMIXDIR/lib:$INFORMIXDIR/lib/esql/$INFORMIXDIR/lib/cli`. The following is an example of environment variable settings on an AIX® system.

   ```
   INFORMIXDIR=/opt/informix/IDS
   LIBPATH=/opt/informix/IDS/lib:/opt/informix/IDS/lib/esql:/opt/informix/IDS/lib/cli
   PATH=/opt/informix/IDS/bin
   ```

4. For the Informix CLI stage, if the data source uses a translation DLL, you must add `INFORMIXDIR/lib/esql` to the shared library search path. If `INFORMIXDIR/lib/esql` is not added, a message that is related to loading translation DLL is logged in the IBM InfoSphere DataStage job log.

Example

The following example shows sample DSN entries for AIX, Solaris, HP, and Linux platforms.

```
[Informix]
Driver=/home/informix/csdk/lib/cli/iclit09b.so
Description=IBM INFORMIX ODBC DRIVER
Database=<stores_demo>
LogonID=<user_id>
Password=<password>
ServerName=<informixserver>
HostName=<informixhost>
Service=<online>
Protocol=ontlitcp EnableInsertCursors=0 GetDBListFromInformix=0
CLIENT_LOCALE=en_us.8859-1
DB_LOCALE=en_us.8859-1
CursorBehavior=0 CancelDetectInterval=0 TrimBlankFromIndexName=1
ApplicationUsingThreads=1 TRANSLATIONDLL=/home/informix/csdk/lib/esql/igo4a304.so
</online></informixhost></informixserver></password></userid></stores_demo>
```

Every Informix data source to which the IBM InfoSphere DataStage jobs connect must have an entry in the `.odbc.ini` file. You must specify values for the Database and Server name properties. The CLIENT_LOCALE and DB_LOCALE fields are optional. If you add login ID (UID) or password (PWD) properties, the User Name and Password properties can be left blank. The values that are specified for the login ID (UID) and password (PWD) properties override the values that are

specified for the User Name and Password properties in the .odbc.ini file.

Configuring the environment for Informix Enterprise stages

You must have the correct privileges and set the environment variables to use the Informix Enterprise stage. You must also have a valid account on the databases to which you connect.

Before you begin

- Install the client libraries. To run jobs with Informix XPS stages on AIX systems, install the Informix client SDK 3.5 (all modifications) with the Informix XPS server.
- Make sure that Informix XPS server is running.

Procedure

1. Set the INFORMIXDIR environment variable to point to the installation directory of the IBM Informix server.
2. Set the INFORMIXSERVER environment variable to point to coserver name of coserver 1 in sqlhosts. Make sure that the coserver is accessible from the node on which you invoke the IBM InfoSphere DataStage job.
3. Set the INFORMIXSQLHOSTS environment variable to point to the sql hosts path. For example, /disk6/informix/informix_runtime/etc/sqlhosts.
4. To run jobs with Informix XPS stages on AIX systems, set the **LIBPATH** environment variable as follows: LIBPATH=$APT_ORCHHOME/lib:$INFORMIXDIR/lib:`dirname $DSHOME`/branded_odbc/lib:$DSHOME/lib:$DSHOME/uvdlls:$DSHOME/java/jre/bin/classic:$DSHOME/java/jre/bin:$INFORMIXDIR/lib:$INFORMIXDIR/lib/cli:$INFORMIXDIR/lib/esql

Configuring access to JDBC data sources

Before you can use the JDBC connector, you must set up the driver configuration file. The connector uses this file to obtain information about the available JDBC drivers in your system.

Procedure

1. Create a driver configuration file named isjdbc.config with read permission enabled for all the users. The driver configuration file name is case sensitive.
2. Open a text editor and include the following two lines that specify the class path and driver Java classes:

   ```
   CLASSPATH=driver_classpath
   CLASS_NAMES=driver_class_names
   ```

   The value driver_classpath is the cumulative Java class path for the JDBC drivers that you plan to utilize through the connector. The value is specified as a semicolon separated list of fully-qualified directory paths, .jar file paths and .zip file paths. For example, if the driver that you plan to use is implemented as a .jar file, include the full path to that .jar file in the driver_classpath value. For more information about the class path requirements for your driver, see the driver documentation.

   The value driver_class_names is a semicolon separated list of fully-qualified driver class names implemented by the JDBC drivers that you plan to utilize through the connector. The driver classes are the Java classes in the drivers that implement the java.sql.Driver JDBC API interface. For more information about the driver class implemented by the driver, see your JDBC driver documentation. Note that you do not need to provide this information for the

drivers implemented as JAR files and based on JDBC 4.0 or later JDBC specification because in those cases the connector is able to automatically determine the driver class name from the driver JAR file.

3. Save the `isjdbc.config` file on the InfoSphere Information Server engine tier host under the directory `IS_HOME/Server/DSEngine` directory, where *IS_HOME* is the InfoSphere Information Server home directory. For example, the home directory might be `C:\IBM\InformationServer` on a Windows system or `/opt/IBM/InformationServer` on a Linux or UNIX system. If the engine tier in your InfoSphere Information Server installation consists of multiple hosts, this file must be available from the same location on all the hosts. You can make this file available from the same location on all the hosts, by configuring the `DSEngine` directory as a shared network directory.

Example

This example shows how you can set up the driver configuration file.

Assume the following details:

- You want to use the JDBC connector with the following two JDBC drivers: JDBC 4.0 driver, Driver A and JDBC 3.0 driver, Driver B.
- Driver A is implemented as the `/opt/productA/driverA.jar` file, and Driver B is implemented as the`/app/productB/driverBimpl.jar` file.
- For Driver A, you determine that the name of the driver Java class is `com.example.A.Driver`, and for Driver B, you determine that the driver Java class name is `com.example.DriverB`.

To set up the driver configuration file, complete the following actions:

1. Create the `isjdbc.config` file and enter the following two lines:

```
CLASSPATH=/opt/productA/driverA.jar;/app/productB/driverBimpl.jar
CLASS_NAMES=com.example.A.Driver;com.example.DriverB
```

   Note: The `com.example.A.Driver` value can be omitted from the `CLASS_NAMES` because the Driver A is a JDBC 4.0 driver, and for this driver the connector can automatically retrieve the driver class name from the driver JAR file.

2. Save the file on the InfoSphere Information Server engine tier host under the `IS_HOME/Server/DSEngine` directory, where *IS_HOME* is the InfoSphere Information Server home directory.

After making changes to the driver configuration file you do not need to restart the DataStage engine, ISF agents or WebSphere Application Server. The JDBC connector recognizes the changes that are made to this file the next time it is used to access JDBC data sources.

Configuring access to Microsoft SQL Server databases

To configure access to Microsoft SQL Server, you must set the **`ODBCINI`** environment variable. You must also ensure that the Microsoft SQL Server database is accessible from the Microsoft SQL Server client and test connectivity between the Microsoft SQL Server client and the Microsoft SQL Server databases.

Before you begin

- Install client libraries.

About this task

When you are connecting to a remote database, ensure that the database server is configured to allow remote connections over the TCP/IP protocol.

The Microsoft SQL Server Client cannot be installed on UNIX. Because of this, the InfoSphere DataStage Dynamic RDBMS plug-in stage on UNIX cannot use the Bulk Insert mode operation when the stage is configured for Microsoft SQL Server database type. The DRS plug-in stage on Windows uses the Microsoft OLE DB API for Bulk Load operations and the API is not available on UNIX. When the DRS plug-in stage is configured for Microsoft SQL Server database on UNIX, the database type for the stage is automatically switched to ODBC.

For more information about configuring access to SQL Server InfoSphere DataStage, see the *IBM InfoSphere DataStage and QualityStage Connectivity Guide for Microsoft SQL Server and OLE DB Data*.

Procedure

1. On UNIX or Linux, set the **ODBCINI** environment variable to point to the `.odbc.ini` file in which the Microsoft SQL Server connection definitions are created.
2. From the Microsoft SQL Server driver on Windows, test if the Microsoft SQL Server database is ready to receive incoming connections:
   a. In the Microsoft SQL Server DSN configuration window, specify the connection information for the Microsoft SQL Server database.
   b. Click **Finish**.
   c. Click **Test Data Source**.
3. On UNIX, to test if the SQL server database is ready to receive incoming connections:
   a. Open the UNIX command-line utility.
   b. Run the DataDirect ODBC example application from the *installation_directory*`/example` directory, where *installation_directory* is the path to the DataDirect ODBC example application installation directory.

```
[root@RH2011 example]# ./example
./example DataDirect Technologies, Inc. ODBC Example Application.

Enter the data source name : mysqlserver

Enter the user name        : username

Enter the password         : passoword

Enter SQL statements (Press ENTER to QUIT)
SQL>
Exiting from the Example ODBC program
[root@RH2011 example]# pwd
installation_directory/example
[root@RH2011 example]#
```

Configuring access to Netezza databases

To configure access to Netezza databases, you must install and configure the Netezza ODBC driver and create the data source.

Configuring access to Netezza databases on Linux and UNIX

To configure access to Netezza databases, you must specify parameters in the `.odbcinst.ini` file to configure the Netezza ODBC driver and also modify the `.odbc.ini` file to configure the data sources.

Before you begin

- Install client libraries.

About this task

If an `.odbcinst.ini` configuration file exists, you can modify the same file. If there is no existing `.odbcinst.ini` configuration file, then you can use the `odbcinst.ini.sample` to create the `.odbcinst.ini` configuration file. In most scenarios, you can use the contents of the `odbcinst.ini.sample` file without any changes. However, in the following scenarios, you must change the configuration file:

- If your client system was configured for ODBC drivers other than the Netezza ODBC driver and you want to continue to use those ODBC drivers, do not modify the existing entries in the `.odbcinst.ini` file. Add an entry for the Netezza ODBC driver at the end of the existing contents of the `.odbcinst.ini` file.
- If the Netezza client software and a Netezza ODBC driver were installed on your client system, check if the Netezza ODBC driver is configured. If it is not, add an entry to the end of the existing contents of the `.odbcinst.ini` file.

If the `.odbc.ini` configuration file exists in your home directory (For example, /home/myname) check if it contains entries for the Netezza appliance data sources to access. If it does not, copy the contents of the `odbc.ini.sample` file to the end of your existing `.odbc.ini` configuration file. Do not modify any existing entries in the file.

If you are using the InfoSphere Information Server version of the `.odbc.ini` configuration file on Linux, create a symbolic link in the folder where the configuration file exists to make sure that the Netezza connector works correctly:

1. Log on as the InfoSphere DataStage administrator.
2. To change to the installation directory of InfoSphere Information Server, enter the command: `cd /opt/IBM/InformationServer/Server/DSEngine` .
3. To create a symbolic link, enter the command: `ln -s .odbc.ini odbc.ini`.

Procedure

1. Log in using your user ID and password.
2. Configure the Netezza ODBC driver:
   a. Copy the contents of the `/usr/local/nz/lib/odbcinst.ini.sample` file.
   b. Modify the configuration entries depending on your requirement. Consult your Netezza system administrator to check if you must modify any specific configuration entries for your installation.
   c. Save the file as `.odbcinst.ini`.
3. Configure the Netezza appliance data sources:
   a. Copy the contents of `odbc.ini.sample` file into your home directory (for example, `/home/myname`) and rename it `.odbc.ini`.

b. Optional: To add the Netezza data sources to an existing `.odbc.ini` file, add the lines after [NZSQL] from the sample file to the existing `.odbc.ini` file. In the [ODBC Data Sources] section, add `NZSQL = NetezzaSQL` to the list of data source names.

c. Save and close the file.

4. Set the environment variables:

```
export ODBCINI=path_to_odbc.ini_file
export NZ_ODBC_INI_PATH=location_of_odbc.ini_file
```

Note: If the Netezza entries were added to an existing `odbc.ini` file, set only the **NZ_ODBC_INI_PATH** variable.

5. To restart the server engine and the ASB Agent, enter the following command:

```
cd Install_directory/Server/DSEngine/bin
./uv -admin -stop
./uv -admin -start
cd Install_directory/ASBNode/bin
. ./NodeAgents_env_DS.sh
./NodeAgents.sh stopAgent
./NodeAgents.sh start
```

Configuring access to Netezza databases on Microsoft Windows

If InfoSphere Information Server runs on the Microsoft Windows operating system, you must create and configure the data source after you install the Netezza ODBC driver.

Before you begin

- Install database client libraries.
- On 64-bit Windows computers, make sure that you run the 32-bit version of the Microsoft ODBC Data Source Administrator `C:\Windows\SysWOW64\odbcad32.exe`, as InfoSphere Information Server is a 32-bit application. If you run the 64-bit version of the ODBC administrator application, the Netezza connector fails to locate the specified data source name. If the ODBC administrator application is not accessible through the File menu by default, use the Windows Explorer to access the application.
  - On 32-bit Windows, the 32-bit driver is installed in the `C:\Windows\System32` directory.
  - On 64-bit Windows, you can install both 32-bit and 64-bit drivers. The 32-bit driver is installed in the `C:\Windows\SysWOW64` directory and 64-bit version is installed in the `C:\Windows\System32` directory.

Procedure

1. Create the data source:
   a. Do one of the following actions depending on your operating system:
      - On a 32-bit Windows system, click **Start** > **Control panel** > **Administrative Tools** > **Data Sources (ODBC)**.
      - On a 64-bit Windows system, use Explorer to navigate to `C:\Windows\SysWOW64\odbcad32.exe`.

   b. On the System DSN page, click **Add**.

   c. On the Create New Data Source page, select **NetezzaSQL** as the driver to set up the data source for, and then click **Finish**.
2. Configure the ODBC driver:

a. On the Netezza ODBC Driver Setup page, specify details about the data source.
b. In the **Server** field, specify the host name or the IP address of the Netezza system to which the ODBC driver connects.
c. To test the connection, specify the username and password, and then click **Test Connection**.

Configuring access to ODBC data sources

There are no special installation requirements for the ODBC connector . However, Before you can use the ODBC connector in a job, you need to configure database drivers, driver managers, and data source names.

Database drivers

You must install and configure database drivers and at least one driver manager before you can use the ODBC connector.

Supported drivers are included with the installation of the product.

The following drivers have limitations or special configuration requirements when you use them with the ODBC connector:

IBM Text File Driver

- Supported data types. Only Numeric, Date, and VarChar are supported.
- Unsupported runtime data types. The following runtime data types are not supported:
  - Bit
  - Binary
  - LongNVarChar
  - LongVarBinary
  - LongVarChar
  - NChar
  - NVarChar
  - Time
  - Timestamp

SQL Server driver

- Large objects (LOBs) by reference. You cannot pass them by reference with this driver. You must transfer them inline.

SQL Server native driver

- Transfer large objects (LOBs) by reference. You cannot pass them by reference with this driver. You must transfer them inline.

SQL Server wire driver

- Transfer large objects (LOBs) by reference. You cannot pass them by reference with this driver. You must transfer them inline.
- View design-time data. To view design-time data that contains spaces, you must select the quoted identifiers check box on the **Advanced** tab of the driver setup window.

Supported data sources

Before the ODBC connector can open a table or file to read and write data, a connection must be defined to the data source that contains that table or file.

To be a supported data source, the ODBC connector must be able to perform read, write, and lookup SQL statements and to exchange data between external data sources and the IBM InfoSphere DataStage data sets. You must also define names for each data sources that are specific to the operating system, driver manager, and driver.

For the ODBC connector, the ODBC driver manager establishes the connection. To establish your data sources, the following requirements must be met:

- The driver and driver manager must be installed on the system where the connector is installed and running. The data source can be on a remote system. You can connect to multiple data sources.
- For parallel jobs, the driver and driver manager must be installed on every node the ODBC connector runs.

You can connect to only one ODBC driver manager at a time.

When you configure the driver manager and you are logged into the IBM InfoSphere DataStage and QualityStage® Designer, you can see a list of data source names in the **Data source** property by clicking the **Data source** button. If you are working in a design environment that is not connected to the server, you can type a value in the **Data source** property.

Configuring the ODBC driver and the ODBC data source name

To use ODBC data sources in a job, you must configure the ODBC driver and the ODBC data source name (DSN) definitions.

Before you begin

- Install client libraries.
- Test the connection to the ODBC data source.
- On 64-bit Windows computers, make sure that you run the 32-bit version of the Microsoft ODBC Data Source Administrator `C:\Windows\SysWOW64\odbcad32.exe`, as InfoSphere Information Server is a 32-bit application. If you run the 64-bit version of the ODBC administrator application, the Netezza connector cannot to locate the specified data source name. If the ODBC administrator application is not accessible through the File menu by default, use the Windows Explorer to access the application.
- For more information, see the topics about configuring ODBC access in the configuring product modules section of the IBM InfoSphere Information Server Planning, Installation, and Configuration Guide.

Procedure

1. Configure the ODBC DSN definitions:

Table 5. Configuring the ODBC data source name definitions

Operating System	Procedure
UNIX or Linux	Set the ODBCINI environment variable to point to the `.odbc.ini` file. The`.odbc.ini` file contains the ODBC DSN definitions. **Note:** The ODBCINI environment variable is set in the `dsenv` script automatically as part of the InfoSphere Information Server installation process.
Microsoft Windows	The DSN definitions are managed by the ODBC driver manager application included with the operating system. The ODBC DSN definitions must be configured as System DSN definitions in the ODBC data source Administrator. The ODBCINI environment variable is not applicable on Microsoft Windows.

2. Add the path to the directory that contains the client libraries to the library path environment variable. The default path for client libraries is as follows:
   - On Windows, `C:\IBM\InformationServer\ODBCDrivers`. On the Microsoft Windows, the ODBC driver manager library is provided by the operating system. The location of the ODBC driver manager is automatically included in the PATH environment variable.
   - On Linux and UNIX, `/opt/IBM/InformationServer/Server/branded_odbc/lib`. The ODBC driver manager is included with InfoSphere Information Server
3. On UNIX and Linux computers, to restart the server engine and the ASB Agent, enter the following command:

```
cd Install_directory/Server/DSEngine/bin
./uv -admin -stop
./uv -admin -start
cd Install_directory/ASBNode/bin
. ./NodeAgents_env_DS.sh
./NodeAgents.sh stopAgent
./NodeAgents.sh start
```

Configuring the ODBC data source in a parallel processing environment

You can configure the ODBC data source in a parallel processing environment with one conductor node and multiple player nodes.

Before you begin

- Install client libraries.

Procedure

1. Add the path to the directory that contains the client libraries to the library path environment variable.
2. Copy the following directory from the conductor node to all computer nodes:
   - On Windows, `C:\IBM\InformationServer\ODBCDrivers`.
   - On Linux and UNIX, `/opt/IBM/InformationServer/Server/branded_odbc/lib`.
3. Copy the `.odbc.ini` file from the conductor node to all computer nodes.

4. On UNIX and Linux computers, to restart the server engine and the ASB Agent, enter the following command:

```
cd Install_directory/Server/DSEngine/bin
./uv -admin -stop
./uv -admin -start
cd Install_directory/ASBNode/bin
. ./NodeAgents_env_DS.sh
./NodeAgents.sh stopAgent
./NodeAgents.sh start
```

Configuring access to Oracle databases

You can configure access to an Oracle database from the Oracle client system by setting environment variables and by updating Oracle configuration files such as `tnsnames.ora` and `sqlnet.ora`. For more information, see the Oracle product documentation.

Before you begin

- Install client libraries.
- Ensure that your system meets the system requirements and that you have a supported version of the Oracle client and Oracle server. For system requirement information, see http://www.ibm.com/software/data/infosphere/info-server/overview/.
- Ensure that the Oracle client can access the Oracle database. To test the connectivity between the Oracle client and Oracle database server, you can use the Oracle SQL*Plus utility.

About this task

You can use the `dsenv` script to update the environment variables that are used to configure access to Oracle databases. If you use the script, you must restart the server engine and the ASB Agent after you update the environment variables.

Procedure

1. Set either the **ORACLE_HOME** or the **TNS_ADMIN** environment variable so that the Oracle connector is able to access the Oracle configuration file, `tnsnames.ora`.
   - If the **ORACLE_HOME** environment variable is specified, then the `tnsnames.ora` file must be in the *$ORACLE_HOME*`/network/admin` directory.
   - If the **TNS_ADMIN** environment variable is specified, then the `tnsnames.ora` file must be in the *$TNS_ADMIN* directory.
   - If both environment variables are specified, then the **TNS_ADMIN** environment variable takes precedence.
   - Setting these environment variables is not mandatory. However, if one or both environment variables are not specified, then you cannot select a connect descriptor name to define the connection to the Oracle database. Instead, when you define the connection, you must provide the complete connect descriptor definition or specify an Oracle Easy Connect string.

   Note: If you use the Oracle Basic Instant Client or the Basic Lite Instant Client, the `tnsnames.ora` file is not automatically created for you. You must manually create the file and save it to a directory. Then specify the location of the file in the **TNS_ADMIN** environment variable. For information about creating the `tnsnames.ora` file manually, see the Oracle documentation.

2. Optional: Set the library path environment variable to include the directory where the Oracle client libraries are located. The default location for client libraries are as follows:
   - On Windows, `C:\app\username\product\11.2.0\client_1\BIN`, where *username* represents a local operating system user name. If the complete Oracle database product is installed on the InfoSphere Information Server engine computer instead of just the Oracle client product, then the path would be `C:\app\username\product\11.2.0\dbhome_1\BIN`.
   - On Linux or UNIX, `u01/app/oracle/product/11.2.0/client_1`
3. Set the **NLS_LANG** environment variable to a value that is compatible with the NLS map name that is specified for the job. The default value for the **NLS_LANG** environment variable is `AMERICAN_AMERICA.US7ASCII`.

   The Oracle client assumes that the data that is exchanged with the stage is encoded according to the **NLS_LANG** setting. However, the data might be encoded according to the NLS map name setting. If the **NLS_LANG** setting and the NLS map name setting are not compatible, data might be corrupted, and invalid values might be stored in the database or retrieved from the database. Ensure that you synchronize the **NLS_LANG** environment variable and NLS map name values that are used for the job.

   On Microsoft Windows installations, if the **NLS_LANG** environment variable is not set, the Oracle client uses the value from the Windows registry.

Configuring access to Sybase databases

To configure access to Sybase databases, you set environment variables on the engine tier computer and specify database information in the interfaces file.

Before you begin

- Install and configure the SQL Server or Sybase client software. The BCPLoad stage uses the BCP API in the DBLIB/CTLIB and NetLIB client libraries. You must ensure that these components are installed on the InfoSphere DataStage server that set up as a client to the SQL Server DBMS. For more information, see DBMS documentation.
- Make sure that the Sybase database server is accessible from the Sybase client.
- Create table in the database on the SQL Server.
- Make sure that the database is registered on the Sybase client.

Procedure

1. Set the **SYBASE** environment variable to point to the Sybase installation directory. For example, `export SYBASE=/disk3/Sybase`.
2. Set the **SYBASE_OCS** environment variable to point to the Sybase Open Client directory. For example, `export SYBASE_OCS=OCS-12_5`. This value indicates the version and release of the Open Client product.
3. Specify the database name, host system name or IP address, and port number in the interfaces file (for example, `sql.ini`) in the `$SYBASE` directory.
4. Set the **DSQUERY** environment variable to the name of the Sybase database server to connect to by default when the server name is not specified in the connection request. If the environment variable is not set, the default value SYBASE is used.
5. Set the PATH and library path environment variable to point to the directory that contains the client libraries.

- On Windows, `%SYBASE%\%SYBASE_OCS%\bin` and `%SYBASE%\%SYBASE_OCS%\dll` directories, where *SYBASE* and *SYBASE_OCS* represent the Sybase product installation home directory and Sybase Open Client directory.
- On Linux and UNIX, `$SYBASE/$SYBASE_OCS/lib`.

Note: Make sure that `$SYBASE/$SYBASE_OCS/bin` is displayed first in the PATH environment variable.

6. For the BCPLoad stages, configure the database for the fast copy (bulk load) option, by using a stored procedure. When this option is used, data is loaded without recording every insert in a log file. For more information about using stored procedures, see *Using Stored Procedures*.
7. Get login privileges to Sybase by using a valid Sybase user name and corresponding password, server name, and database. These must be recognized by Sybase before you attempt to access it.
8. Use the `dsedit` utility that is provided with the Sybase Open Client to configure connection to the Sybase database.
9. Test the connectivity to Sybase database outside of InfoSphere DataStage by using tools such as `$SYBASE/$SYBASE_OCS/bin/isql` tool in Sybase Open Client.
10. Complete the following steps to access Sybase databases with NLS in Sybase enterprise stages.
    a. Create a database and configure the language of your choice. For example, create database <<database path>> COLLATION 932JPN for a Japanese (shift_jis) database.
    b. Install the IBM InfoSphere DataStage server in that particular language, for example, Japanese (shift_jis).
    c. To set the language for InfoSphere DataStage client, use the **NLS** tab in job properties to select the language.
    d. Make sure that the selected language is set as default in the operating system of the system on which the InfoSphere DataStage client is installed.

Permissions required to access Sybase databases

To complete operations on tables that are hosted by Sybase ASE and Sybase IQ databases, you require specific privileges on the table.

Table 6. Permissions required to access Sybase databases.

You require write, read, upsert and lookup privileges on the table to complete operations on tables that are hosted by Sybase ASE and Sybase IQ databases.

Operation	Options	Syspartition (only for Sybase ASE)	sysobjects	SELECT privilege on table	INSERT privilege on table	Delete table	Create table
Write	Create/ Replace	Yes	Yes	No	No	No	Yes
Write	Append	Yes	Yes	Yes	Yes	No	No
Write	Truncate	Yes	Yes	Yes	Yes	Yes	No
Read	All	No	Yes	Yes	No	No	No

Table 6. Permissions required to access Sybase databases (continued).

You require write, read, upsert and lookup privileges on the table to complete operations on tables that are hosted by Sybase ASE and Sybase IQ databases.

Operation	Options	Syspartition (only for Sybase ASE)	sysobjects	SELECT privilege on table	INSERT privilege on table	Delete table	Create table
Upsert	Update/ Insert	• UPDATE privilege on the table that you want to update. • INSERT privilege on the table into which you want to insert records.	• UPDATE privilege on the table that you want to update. • INSERT privilege on the table into which you want to insert records.	• UPDATE privilege on the table that you want to update. • INSERT privilege on the table into which you want to insert records.	• UPDATE privilege on the table that you want to update. • INSERT privilege on the table into which you want to insert records.	• UPDATE privilege on the table that you want to update. • INSERT privilege on the table into which you want to insert records.	• UPDATE privilege on the table that you want to update. • INSERT privilege on the table into which you want to insert records.
Lookup	All	No	Yes	Yes	No	No	No

Configuring access to Teradata databases

To configure access to Teradata databases, you must install Teradata tools and Teradata transporters and set the environment variables.

Before you begin

Install database client libraries.

Procedure

1. Install Teradata tools and utilities on all nodes that run parallel jobs. For more information, see the installation instructions in the Teradata product documentation.
2. Install the Teradata Parallel Transporter. For more information, see the installation instructions in the Teradata product documentation.
3. Set the environment variables.

Table 7. Required Environment variables

Operating System	Environment variables

AIX	`TWB_ROOT=/usr/tbuild/08.01.00.02`
	`PATH=$TWB_ROOT/bin:$PATH`
	`LIBPATH=$TWB_ROOT/lib:$LIBPATH`
	`NLSPATH=$TWB_ROOT/msg/%N`
	`export TWB_ROOT PATH LIBPATH NLSPATH`

Table 7. Required Environment variables (continued)

Operating System	Environment variables

HP-UX	`TWB_ROOT=/usr/tbuild/08.01.00.02`
	`PATH=$TWB_ROOT/bin:$PATH`
	`SHLIB_PATH=$TWB_ROOT/lib:$SHLIB_PATH`
	`NLSPATH=$TWB_ROOT/msg/%N`
	`export TWB_ROOT PATH SHLIB_PATH NLSPATH`

Solaris	`TWB_ROOT=/usr/tbuild/08.01.00.02`
	`PATH=$TWB_ROOT/bin:$PATH`
	`LD_LIBRARY_PATH=$TWB_ROOT/lib:$LD_`
	`LIBRARY_PATH NLSPATH=$TWB_ROOT`
	`/msg/%N export TWB_ROOT PATH LD_`
	`LIBRARY_PATH NLSPATH`

Testing database connections by using the ISA Lite tool

After you establish connection to the databases, test the database connection by running the IBM Support Assistant (ISA) Lite for InfoSphere Information Server tool.

For more information about the ISA Lite tool, see the topic about installation verification and troubleshooting.

Setting the library path environment variable

To apply an environment variable to all jobs in a project, define the environment variable in the InfoSphere DataStage and QualityStage Administrator. The values that are specified for the library path and path environment variables at the project or job level are appended to the existing system values for these variables.

About this task

For example, suppose that directory `/opt/branded_odbc/lib` is specified as the value for the library path environment variable at the project level. Directory `/opt/IBM/InformationServer/Server/branded_odbc/lib`, which contains the same libraries but in a different location is already in the library path that is defined at the operating system level or the `dsenv` script. In this case, the libraries from directory `/opt/IBM/InformationServer/Server/branded_odbc/lib` are loaded when the job runs because this directory appears before directory `/opt/branded_odbc/lib` in the values that are defined for the library path environment variable.

The name of the library path environment variable depends on your operating system.

Operating system	Library path environment variable
Microsoft Windows	**`PATH`**
HP-UX	**`SHLIB_PATH`**
IBM AIX	**`LIBPATH`**
Other supported Linux and UNIX operating systems, and HP-IA	**`LD_LIBRARY_PATH`**

On Linux or UNIX operating systems, the environment variables can be specified in the `dsenv` script. InfoSphere Information Server installations on Windows

operating system do not include the dsenv script.

Setting the library path environment variable in the dsenv file

On Linux or UNIX operating systems, you can specify the library path environment variables in the dsenv script. When environment variables are specified in the dsenv script, they apply to all InfoSphere DataStage projects that run under the InfoSphere Information Server engine.

Before you begin

Install the client libraries.

Procedure

1. Log in as the DataStage administrator user (dsadm if you installed with the default name).
2. Back up the *IS_install_path*/Server/DSEngine/dsenv script. *IS_install_path* is the InfoSphere Information Server installation directory (/opt/IBM/InformationServer if you installed with the default path).
3. Open the dsenv script.
4. Add the path to the directory that contains the client libraries to the library path environment variable.
5. Set up your environment with the updated dsenv file.

   ```
   . ./dsenv
   ```
6. Restart the InfoSphere Information Server engine by entering the following commands:

   ```
   bin/uv –admin –stop
   bin/uv –admin –start
   ```
7. Assume root user privileges, directly with the **su** command or through the **sudo** command if the DataStage administrator user is part of the sudoers list.

   ```
   sudo su - root
   ```
8. Change to the ASB Agent home directory by entering the following commands:

   ```
   cd Install_directory/ASBNode/bin
   ```
9. Restart the ASB Agent processes by entering the following commands:

   ```
   ./NodeAgents.sh stopAgent
   ./NodeAgents.sh start
   ```

Results

After you restart the ASB Agent process, the InfoSphere Information Server services take approximately a minute to register the event.

Setting the library path environment variable in Windows

On the Windows operating system, both the library path and **PATH** environment variables are represented by the **PATH** system environment variable. For InfoSphere Information Server engine and ASB Agent processes to detect changes in the environment variables, the changes must be made at the system level and the InfoSphere Information Server engine must be restarted.

Before you begin

Install the client libraries.

Procedure

1. To edit the **PATH** system environment variable, click **Environment Variable** in **Advance System Settings**, and then select **PATH**.
2. Click **Edit**, then specify the path to the directory containing the client libraries.
3. Click OK.
4. Restart the InfoSphere Information Server engine.
5. Restart the ASB Agent processes.

Chapter 2. DRS Connector stage

When you use InfoSphere DataStage to access various types of relational databases, you can choose from a collection of connectivity options. For new jobs, use the DRS Connector stage, which offers better functionality and performance.

A connector is a component that provides data connectivity and metadata integration for external data sources, such as relational databases or messaging software. A connector typically includes a stage that is specific to the external data source.

The DRS Connector stage is one of several different stages that access various types of relational databases. In addition to the DRS Connector stage, you can use the Dynamic RDBMS stage.

If you have jobs that use the older stages and want to use the connectors, use the Connector Migration Tool to migrate jobs to use connectors.

Overview

Use the DRS Connector stage to access relational database management systems by using the native interfaces that are available for the corresponding databases. The choice of the database type is not determined when the stage is placed on the job canvas but is instead specified when the stage is configured.

In addition, an InfoSphere DataStage job parameter can be used for the database type setting in the stage. In that case, the selection of the database type is made when the job runs.

The DRS Connector stage is similar to ODBC connectivity stages in that a single stage can be configured to access many different database types. An important difference is that the DRS Connector stage accesses databases directly through the native database client libraries, while ODBC Stages access databases through the ODBC drivers.

In cases when the dynamic database type selection support is not required, such as when the database type choice is firmly determined in the current environment and is unlikely to change, it is recommended to use the native connector stage for the database type in question instead of using the DRS Connector stage. For example when IBM DB2 is the database type of choice then it is recommended to use DB2 Connector stage instead of the DRS Connector stage.

The use of native connector stages instead of the DRS Connector stage allows you to take advantage of various features that are not supported by the DRS Connector but are supported by the native stages. For example, for IBM DB2 and Oracle database types there are many features in the areas of partitioning and parallelism that are supported by the corresponding native connector stages and are not supported by the DRS Connector stage.

The DRS Connector stage supports the following features:

- Database read, write and lookup operations.

- Automatic generation of SQL statements at runtime based on the specified table name, column definitions on the link and other user defined properties of the stage. The SQL statements can also be manually specified.
- Running an SQL statement specified in a file that resides on the engine tier.
- Automatically creating or replacing target tables in the database during the job runtime.
- Automatically clearing the contents of the existing target table at runtime before writing data to it.
- Writing rows to the database and reading rows from the database in batch (array) mode using an array of the specified size.
- Specifying a transaction size (number of records per transaction).
- Specifying an isolation level for the transactions.
- Writing data to the database in bulk load mode using native bulk load interfaces provided by the database.
- Rejecting bad records (server jobs only). Requires the use of the Transformer stage.
- Record ordering across multiple input links (in parallel jobs only).
- Running user-specified SQL statements on the database after the job starts and before any records are actually processed by the stage. The statements can be specified directly in the stage or in a file that resides on the engine tier.
- Running user-specified SQL statements on the database after all the records have been processed by the stage and before the job ends. The statements can be specified directly in the stage or in a file that resides on the engine tier.
- Derivation attributes for the columns on the output link of the stage. The stage uses derivation values instead of column names when it generates `SELECT` statements to use at runtime for reading data from the database.
- Handling the specified table name and column names as case-sensitive values or case-insensitive values.

Configuration

The DRS Connector stage supports IBM DB2, Oracle, and ODBC data sources. For other database types, you can configure the DRS Connector stage to use the ODBC database type and access the databases through the ODBC drivers that are included with InfoSphere Information Server.

The DRS Connector stage accesses databases directly through the native database client libraries. To connect to the database server when the job runs, the stage needs to be able to locate and load database client libraries installed on the engine tier computer. You must set the library path environment variable to include the directory where the database client libraries reside. For more information on configuring data sources, see the section on configuring access to data sources.

Settings for the DRS Connector stage

After the DRS Connector stage is placed on the job canvas, it needs to be configured to perform the operation intended by the job design.

Double-click the stage to open the **Stage** dialog box. You can set the properties on the **Stage** dialog box to the appropriate values depending on what action is expected to be performed by the stage when the job runs.

DRS Connector stage settings

In the DRS Connector stage, some properties are displayed on the main stage dialog itself and some are located under separate tabs.

If the stage has multiple links, selecting a link will display only the Usage properties for that link. If the stage has a single link, selecting the link will display the Usage properties for that link as well as the Connection properties for the stage.

The Stage properties dialog box contains the following fields:

General tab
Provides an optional description field as a free-form text.

Properties tab
This tab organizes the properties into a tree structure. Displays the properties in the tree structure divided in two sections: Connection and Usage. In some cases the properties appear directly under the Usage and Connection sections and in some cases are they are located under separate categories.

Connection
The properties in the Connection section apply to the stage itself and all the links of the stage.

Usage The properties in the Usage section in most cases apply to individual links. The **Record ordering** property is used to record ordering feature. This property is defined at the stage level and is therefore shared across all the links defined for the stage.

This tab also has the following hyperlinks:

Test Attempts to establish a connection to the database using the currently defined Connection properties. If the connection is established a message box indicating successful operation is displayed. Otherwise a message box is displayed with the error details.

Load Loads a DRS Connector connection definition which was previously saved to the metadata repository.

Save Saves the current connection definition (represented by the values of the properties in the **Connection** section) to the common repository so that it can be reused by other DRS Connector stages.

Advanced tab
Displayed only for stages on the parallel job canvas. It provides properties that specify whether to run the stage in sequential or parallel mode. If the stage is running in parallel mode this tab also provides the options to constrain the stage to run on a specific subset of processing nodes defined in the parallel engine configuration file (`default.apt` file by default).

DRS Connector link settings

On the Input tab, you can specify values for the DRS Connector link settings.

On the Input tab, the following tabs are displayed.

General
Displays an optional field for description in free-form text for the link.

Properties
Displays the properties for the selected link. The example shows properties that can be set for an input link.

Link Ordering
Orders links for the record ordering functionality.

Columns
Defines columns on the link. The columns can be defined manually, or a table definition from the repository can be applied on the link.

Advanced
Displayed only for the stages on parallel canvas and can be used to specify buffering strategy for the records.

Partitioning
Displayed only for input links of the stages on the parallel canvas and can be used to specify how records are partitioned across processing nodes on which the stage runs when configured to run in parallel mode.

Using job parameters for stage and link properties

You can set job parameters as environment variables or at the job level.

One option to consider for taking full advantage of the dynamic selection support is to define an environment variable for the database type at the project level and specify the default value that matches the database type currently in use. Then, add this environment variable as a job parameter to each job that uses the stage and specify the environment variable as the database type value in the stage. When at a later point a decision is made to utilize the same jobs to connect to a different database type, you would only need to change the default value of the environment variable at the project level and all the jobs will start using the new database type when they run the next time.

In addition to adding the environment variable as a job parameter, it is also possible to define job parameters directly at the job level. In either case it is possible to specify the default value to be used for the job. And when the job is started from InfoSphere DataStage Director, the value for the job parameter can be specified at that time.

In the DRS Connector stage a job parameter can also be used for Variant selection. For example, an environment variable can be defined at the project level for the Oracle variant. The environment variable is included as job parameter to the jobs that are using the DRS Connector and specified as the Variant property value in the stages. The environment variable default value is `10`. When upgrading from the Oracle client version 10gR2 to version 11gR2, you change the value of environment variable from 10 to 11 at the project level and the jobs will start using variant 11 of the DRS Connector stage the next time they run.

When job parameters are used to specify the database type, then typically it can also be necessary to use job parameters for the database name, username, and password properties since these values will often not be the same for databases of different database types.

Defining a DRS Connector stage connection to the database

When a new DRS Connector stage is added to the server or parallel canvas, you must specify the connection information for the database that the stage connects to at run time.

About this task

When the job starts, the stage connects to the specified database by using the specified database credentials. As the job runs, the stage reads data rows from the database or writes data rows to it, depending on whether output or input links are defined for the stage, respectively.

Procedure

1. Select the relevant database type from the **Connector** drop-down list.
2. Select the relevant database client version. For example, if an Oracle database type is selected, and an Oracle 11g version is installed on the InfoSphere DataStage Server engine machine, the Variant value must be set to 11. If the Oracle 10g client is installed, the Variant value must be set to 10. Some database types have only one variant. If you specified job parameter for the database type property, then you can select the variant for all supported database types, because the actual selection of database type to use at runtime is done at a later point.
3. Set the **Connection string** property value. This value identifies the specific database that you want to connect to. The database client environment needs to be set up properly to resolve this value and to connect to the corresponding database server.
4. Set the **Username** property to the value that identifies the user name for authentication and authorization with the database.
5. Set the **Password** property to the password value of the specified user name. The password is displayed and stored by the stage in encrypted form.
6. Optional: You can set the environment variables by using the Administrator Client, to appear as default variant values for respective connector as follows: `CC_DRSCONNECTOR_DEFAULT_CONNECTION_VARIANT_IBM__DB2VARIANT`, for the DB2 connector, `CC_DRSCONNECTOR_DEFAULT_CONNECTION_VARIANT_ORACLEVARIANT`, for the Oracle connector, and `CC_DRSCONNECTOR_DEFAULT_CONNECTION_VARIANT_ODBCVARIANT`, for the ODBC connector.

   Make sure that all parameters are specified in capital letters.

   Note: These values are effective only when the connector stage editor is opened for the first time.

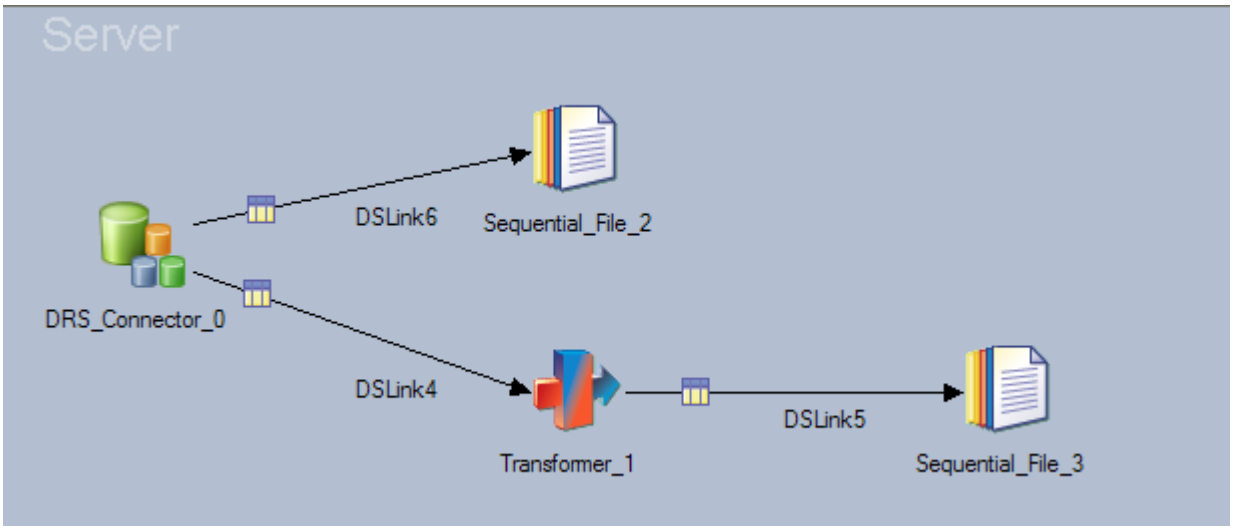
Reading data from the database

To configure the stage to read data from the database, define one or more output links on the stage and configure each link with the query to run on the database.

About this task

When the job runs, for each output link the stage runs the query specified on the link, fetches the result rows from the database, and delivers them to the downstream stage in the job.

The following figure shows the DRS Connector stage (**DRS_Connector_0**) in a server job configured to read data from a database and write that data to two files. To one of the files (**Sequential_File_2**) the data is written directly and to the other one (**Sequential_File_3**) the data is written after being transformed in the Transformer stage (**Transformer_1**).



The DRS Connector stage supports multiple output links on the server canvas, but supports only one output link on the parallel canvas.

When multiple output links are defined for the stage all the links read data from the same database. The database connection definition applies to the stage as a whole and cannot be specified separately for individual links.

Auto-generated SELECT statements

You can configure the stage to automatically generate `SELECT` statements at run time.

For the DRS Connector to automatically generate the statement, you must set the **Read mode** property in the **Usage** section to value `Select rows`.

The following values need to be specified to generate the `SELECT` statement:

- Table name
- Column names on the output link
- `WHERE` clause (optional)
- Other clauses (optional)

On the DRS Connector stage the table name is specified in **Table name** usage property.

If only the table name values are specified, the schema under which the tables reside is the default schema for the user name defined for the stage. To specify a different schema, the table names need to be entered in fully-qualified form such as *schema_name*. *table_name*.

To treat the specified table names as case-sensitive values, they need to be enclosed in quoted identifiers of the target database. Typically double quotation marks are used a quoted identifiers.

The DRS Connector stage provides an option to automatically put quoted identifiers around the table names at runtime by setting the **Enable quoted identifiers** property in the **Usage** section to `Yes`. By default, this property is set to `No`.

The column names on the output link are used by the stage to produce the comma-separated list of columns in the generated `SELECT` statement. The names of the columns on the link are copied to the generated statement text. For each column you can specify the text to use in the auto-generated text instead of the column name. For example, this can be necessary when the column names in the table contain characters that cannot be displayed in the columns grid of the stage, to denote the table to which the column belongs when selecting from multiple tables or to preserve case sensitivity of the column names. The value is specified in the **Derivation** attribute for the column. This attribute is available for all column definitions on the **Columns** tab.

To preserve the case-sensitivity of the columns in the auto-generated statement, you can use the Derivation attribute for the columns and specify column names enclosed in quoted identifiers of the target table. Alternatively, you configure the stage to automatically add quoted identifiers to the column names by setting the **Enable quoted identifiers** property in the **Usage** section to `Yes`.

You can specify the `WHERE` clause to be appended to the auto-generated statement. This clause specifies the condition or filter to apply for producing the result data set. The `WHERE` clause is specified through the **WHERE clause** property located under the **SQL** usage property. The `WHERE` keyword can be included in the specified value but this is not required.

You can also specify additional clauses (such as `ORDER BY`, `GROUP BY` and `HAVING` clauses) to be appended to the auto-generated statement. These clauses specify the sort order and grouping of rows in the result data set. The additional clauses are specified through the **Other clauses** property located under the **SQL usage** property. The keywords that identify the clause types (such as `ORDER BY`, `GROUP BY`, and `HAVING`) must be included with the specified value.

The DRS Connector generates the `SELECT` statement at runtime. The statement text is not displayed in the stage dialog box. It is displayed in the job log when the job starts.

User-defined SELECT statements

You can manually specify a `SELECT` statement to run in the job.

The statement can be specified directly in the stage dialog box, or it can be specified in a file located on the InfoSphere DataStage Server engine machine and referenced by the stage.

To manually specify `SELECT` statement text in the DRS Connector stage you must set **Read mode** property in the **Usage** section to value **User-defined SQL**. The statement text can then be provided through the **Statement** property in the **Usage** section.

The columns in the result set of the SELECT statement need to match the names of the columns on the output link for the stage. For example, if the SELECT statement retrieves the data from a database table with columns C1 and C2, the output link of the stage must also include columns C1 and C2.

When the statement text is stored in a file on the InfoSphere DataStage Server machine, the full path to the file can be specified in the stage and then the stage will open the file at runtime and reads the statement from the file.

To specify the file with the SELECT statement in DRS Connector stage you must set the **Read mode** property in the **Usage** section to **User-defined SQL file**. The file path can then be provided through the **Statement** property in the **Usage** section. If the **FILE=** or **{FILE}** prefix is specified for the **Statement** property in the DRS Connector stage, then the stage treats the remainder of the value as the path to the file that contains the SELECT statement. This is true if the **Read** mode property is set to User-defined SQL file, but also when it is set to User-defined SQL.

Array read mode

When the stage runs the query at run time, it fetches all the rows in the result data set and delivers them on the output link.

You can specify how many rows to fetch at a time from the database. When more than one row is fetched at a time that operation is referred to as batch or array reads.

The array size is specified through the **Array size** property located under the **Session** property in the **Usage** property section. The allowed values for the property are 1 to 999,999,999. The default value is 2,000. When the value is set to 1, the batch or array read mode is effectively disabled and the stage fetches one row at the time from the database.

Writing data to the database

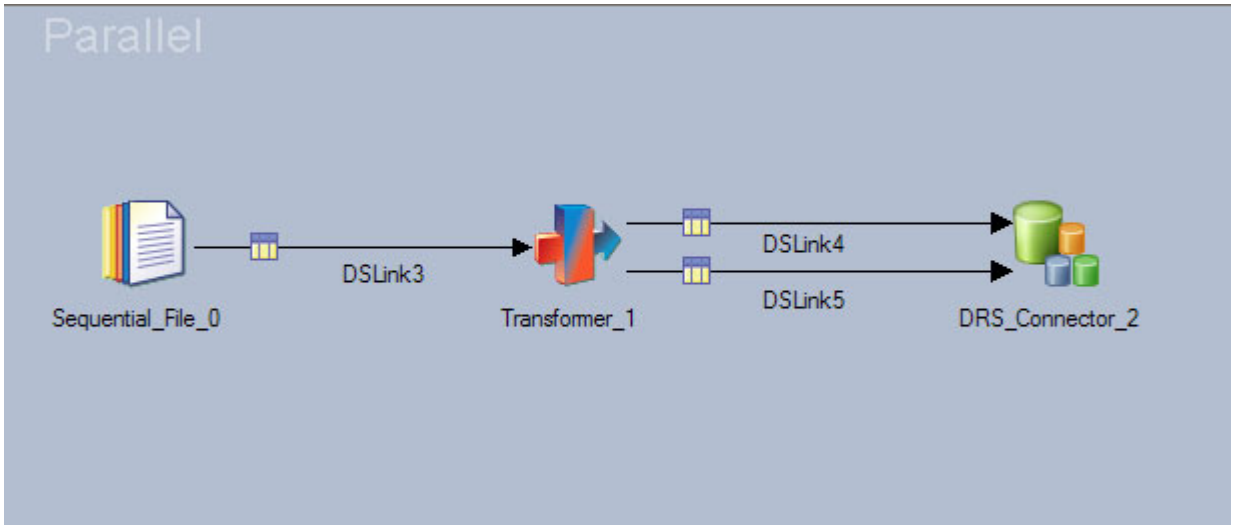
To configure the stage to write data to the database, you must define one or more input links on the stage and configure each link to perform the required write operation on the database.

About this task

When the job runs, for each input link the stage prepares the rows that it receives from the upstream stage in the job and writes them to the database.

When multiple input links are defined for the stage, they all must write data to the same database. The database connection definition applies to the stage as a whole and cannot be specified for individual links.

The following figure shows the DRS Connector stage (**DRS_Connector_2**) in a parallel job configured to write data to a database. The data records are read from a file (**Sequential_File_0**) and are passed to the Transformer stage (**Transformer_1**). The transformed data is then passed in two different formats on two separate links to the DRS Connector stage which can then write it to two different tables for example.



Write modes

The stage supports many different write modes that determine how the write operation is performed on the database.

The write mode is specified in the **Write mode** property in the **Usage** section.

For certain write modes the stage needs to locate the matching rows in the target table for the input rows that it receives on the input link. A row in the target table is considered a match for the input row when it has the same values as the input row for the columns marked as Key column on the link. At least one column on the input link must be marked as a Key column for these write modes.

The following write modes are supported:

Insert rows
The rows are inserted in the table while preserving the existing content of the table. This mode is the default write mode.

Delete rows only
The stage deletes the rows in the target table that match the rows the stage receives on the input link.

Replace existing rows completely
The stage deletes all the rows in the target table that match the rows the stage received on the input link and inserts the rows from the link to the table.

Update existing rows
The stage updates all the rows in the target table that match the rows the stage received on the input link. The values for the columns not marked as Key columns on the link are updated in the matched rows in the table with the value from the input row. The input rows for which no match is detected in the table are ignored.

Update existing rows or insert new ones
This mode is the same as the Update mode except the input rows for which no match is detected in the table are not ignored but are instead inserted to the table. The difference between this mode and the **Replace existing rows completely mode** is for the scenario when input row results

in multiple matches in the table. This is possible since the columns marked as Key columns on the link do not necessarily need to correspond to an actual primary key or unique constraint in the table. When multiple matches are detected for an input row, the **Replace existing rows completely mode** mode will replace all the matching rows with that single input row while this mode will result in updating all the existing rows with values from the input row.

Insert new rows or update existing ones
For each input row the stage first tries to insert it into the database. If the insert operation fails due to unique constraint violation, the connector performs the update operation so all the rows in the table that match the input row are updated.

Insert new rows only
Is the same as the **Insert new rows or update existing ones** mode except the connector does not perform the followup update operation for the rows for which insert resulted in unique constraint violation.

Delete all rows
The entire content of the table is erased and all input rows are ignored. Therefore jobs that contain the stage configured with this mode typically ensure that no records or few records at most are actually delivered to the stage since they are ignored anyway.

Bulk insert
The stage takes advantage of the native bulk load support provided by the backend database to perform loading of rows into the table. When bulk load interface is supported by the database, it is typically faster to write data to the database using that interface than running standard `INSERT` SQL statements.

This write mode is supported only for database types that provide bulk load interface. It is not supported for ODBC database type.

Note: Some ODBC drivers can be configured to write data to the database in bulk load mode.

User-defined SQL
The stage runs the SQL statement or set of statements that is specified by the user.

User-defined SQL file
Is the same as **User-defined SQL** mode except the statements are not specified directly in the stage but in a file located on the InfoSphere DataStage Server machine.

When the stage automatically creates SQL statements for writing rows to the database, it references the specified table name and column names in the generated statements. Many databases treat the table and column names in the statements as uppercase values even if the actual values use mixed cases. To enforce the preservation of upper and lower cases in the specified table and column values, you can configure the stage to automatically enclose these values in quoted identifier characters of the database before inserting them in the generated statements. The quoted identifier character is typically the double-quote character.

To preserve case in the specified table and column names, you can set the **Enable quoted identifiers** property in the **Usage** section to `Yes`. The default value is `No`.

User-defined SQL statements for writing data to the database

In some cases, the INSERT, UPDATE and DELETE statements that are generated automatically might not be suitable for the write operation that you want the connector to complete on the database. In these cases, you can enter the INSERT, UPDATE and DELETE statements manually.

To configure the DRS Connector stage to run user-specified SQL statements instead of auto-generated statements you must set the **Write mode** property in the **Usage** section to **User-defined SQL**.

In the DRS Connector stage the statements need to be specified in the **Statement** property in the **SQL** category in the **Usage** section.

The specified statements need to use bind parameters that the stage associates with the columns on the input link. The bind parameters are specified as question marks. The stage then internally translates the question marks to comply with the bind parameter syntax used by the backend database. When multiple statements are specified, the association between question marks and link columns is performed separately for each statement.

The DRS Connector stage also supports InfoSphere DataStage syntax for bind parameters which makes use of the ORCHESTRATE keyword to denote bind parameters. In this case the parameters directly reference columns on the link. Each parameter is specified in form ORCHESTRATE. *param_name* where *param_name* corresponds to the column on the link that should be associated with that parameter.

You can specify the statements in one of the following combinations:

A single INSERT statement
: The stage associates question mark parameters in the statement with the columns on the link in the same order. The first question mark is associated with first column on the link, the second question mark with the second column on the link and so forth. In this case the stage operates in just like it was configured in **Insert** mode except the INSERT statement is specified manually.

A single UPDATE statement
: The stage associates question marks in the `WHERE` clause of the statement with the columns on the link marked as Key columns. The first question mark in the `WHERE` clause is associated with the first key column on the link, the second question mark with the second key column and so forth. The question marks in the SET clause of the statement are associated with columns on the link that are not marked as Key columns. Again the first question mark in the SET clause is set with the first non-Key column on the link, the second question mark with the second non-Key column and so forth. In this example, the stage operates similarly to **Update matching rows** mode except the UPDATE statement is specified manually.

A single DELETE statement
: The stage associates question marks in the `WHERE` clause of the statement with the columns on the link marked as Key columns. The first question mark in the `WHERE` clause is associated with the first key column on the link, the second question mark with the second key column and so forth. The link columns that are not marked as Key columns are ignored. In this example, the stage operates just like it was configured in **Delete matching rows** mode except the DELETE statement is specified manually.

An INSERT statement followed by an UPDATE statement
The statements are separated by a semicolon character. In this case, the stage operates just like it was configured in **Insert then update** write mode except the statements are specified manually.

An UPDATE statement followed by an INSERT statement
The statements are separated by a semicolon character. In this case, the stage operates like it was configured in **Update then insert** write mode except the statements are specified manually.

A DELETE statement followed by an INSERT statement
The statements are separated by a semicolon character. In this case, the stage operates like it was set in **Delete then insert** write mode except the statements are specified manually.

Any combination of statements
The stage runs each statement separately.

However, if the DRS Connector stage is configured for the Oracle database type, this combination is supported, but the stage does not run the statements individually. Instead the stage runs the statements as a single anonymous PL/SQL block on every input row. Also in this case array write mode is disabled and the input rows are sent to the database one by one where the PL/SQL block statement runs for them.

In addition to specifying SQL statement directly in the stage, it is possible to store statements in a file on the InfoSphere DataStage Server machine and configure the stage to reference this file. The full path to the file is specified in the stage.

To configure the DRS Connector stage to read statements from a file you must set the **Write mode** property in the **Usage** section to **User-defined SQL file**. The file path can then be provided through the **Statement** property in the **Usage** section to `User-defined SQL file`. If a `FILE=` or `{FILE}` prefix is specified for the **Statement** property in the DRS Connector stage, then the stage treats the remainder of the value as the path to the file that contains the user-defined SQL statements. This is true if the **Write mode** property in DRS Connector is set to **User-defined SQL file**, but also when it is set to `User-defined SQL`.

Array write mode

The stage can be configured to send input rows to the database in array (batch) mode.

In this case the rows are not sent to the database one by one but are buffered by the stage in an array structure in memory and when the array is filled out (or there are no more rows on input) the whole array is sent to the database at once. You can specify the size of the array.

In the DRS Connector stage the write array size is specified in the **Array size** property under the **Session** category in the **Usage** section. The default value is `2,000`. The value must be between 1 and `999,999,999`. When set to 1, the array write mode is effectively disabled and rows are sent to the database one by one.

Bulk load write mode

You can use the stage to load data to the target database by using native bulk load interfaces that are provided by the databases.

As the bulk load interfaces differ significantly among the databases, the properties used to configure the load operation also differ significantly.

When bulk load mode is selected for the stage, and a job parameter is used for the database type, the stage interface allows you to specify load properties for all database types for which bulk load is supported. When the job starts and you specify the actual database type to be used, the bulk load properties for that database type become effective and the bulk load properties for the remaining database types are ignored. That way you can take advantage of proprietary bulk load interfaces in different database types while keeping the option to delay the decision which database type to use until the job is actually started.

The DRS Connector stage cannot bulk load data to Informix, Microsoft SQL Server, or Sybase databases. However, you can configure the DRS Connector stage to access Informix through ODBC, and the ODBC DSN can be configured to write data to the table by using bulk load mode.

When the stage is configured to create data and control (configuration) files for the load operation, those files are created in the directory that is specified in the **Work directory** property. In DRS Connector stage, this property is located under the **Bulk load options** category in the **Usage** section.

Loading data to IBM DB2

These are the properties that can be set to customize the bulk load operation when the stage is configured for the IBM DB2 database type.

Load method

Specifies the mechanism to use for providing data to the database to be loaded to the target table. The following values are supported

Sequential file

The connector stores data to a file from which the loader loads it to the database

Named pipe

Default. The connector delivers the data directly to the loader.

Load immediate

Specifies whether to load the data as the job runs or to store data to a file to be loaded later. The allowed values are `Yes` and `No`. The default is Yes which means to load data during the job runtime.

Remove intermediate data file

Specifies whether to remove the data file after completing the load operation. This property is applicable only when the **Load method** property is set to **Sequential file**.

The supported values are `Yes` and `No`.

The default value is `Yes`, meaning the data file is removed after the data has been loaded.

Data format

Specifies the data format to use in the data files when the **Load method** property is set to **Sequential file**.

The supported values are:

Non-delimited ASCII format

ASCII data file with values specified in fixed size format.

This value is the default value.

Delimited ASCII format
ASCII data file with delimiter used as the value separator.

LOB path
Specifies the path to the directory with data files that contain LOB values that need to be loaded to the target table. There is no default value.

File type modifier options
Specifies the modifier options (as free-form text) to be provided for the MODIFY load parameter.

No default value is used for the stage. The default is an empty string.

Load without prompting
Specifies whether the list of specified data files contains all the files that need to be loaded and that the devices or directories listed are sufficient for the entire load operation. The supported values are `Yes` and `No`.

The default value is `Yes`, which indicates that the provided list of data files is sufficient to complete the load operation.

Load mode
specifies the mode in which to load the data.

The supported values are:

Insert Appends new data to the table without changing the existing data in the table.

Replace
Deletes all the data from the table before loading the new data.

This value is the default value.

Restart
Restarts the load operation following the interruption of the previous load operation.

Terminate
Terminates the load operation that was previously interrupted.

Save count
Specifies the number of rows to load before establishing consistency point. The default value 0 is used to specify that no consistency points are established, unless determined to be necessary by the load utility.

Row count
Specifies the total number of initial rows to load. The default value 0 specifies that all the rows in the input data file should be loaded.

Restart count
Specifies the number of rows to skip before starting to load rows. This property should be used when the previous load attempt failed with some records committed to the target table.

The default value 0 specifies that not records should be skipped and to start loading from the first row.

Restart phase
Specifies the phase at which to restart the load operation. Build and Delete values should not be specified if Insert or Replace values are specified for the Load mode property.

The supported values are:

Load Start the operation in the Load phase.

Build Start the operation in the Build phase.

Delete Start the operation in Delete phase.

Warning count
Specifies the number of warnings after which to stop the load operation. The default value is 0 which means the load operation proceeds regardless of the number of warnings that were reported.

Indexing mode
Specifies whether to rebuild indexes or to extend them incrementally. The supported values are:

Auto-select
The load utility automatically chooses the most optimal option.

This value is the default value.

Rebuild
The indexes are rebuilt after the load

Incremental
The indexes are maintained (incrementally built) during the load

Deferred
The decision is deferred

Load message file name
Specifies the path to the local file to use for storing warning and error messages during the load operation.

There is no default value.

Temporary files directory
Specifies the path to the directory that DB2 uses for storing temporary files that it creates for the load operation.

Exception table name
Specifies the name of the table to which to insert rows in error.

There is no default value.

Statistics
The types of statistics to collect for the table.

The collection of statistics is not supported if the **Load mode** property is set to **Insert** or **Replace**.

The supported values are:

Table statistics
Table statistics are gathered.

Table and index statistics
Table and index statistics are gathered.

Index statistics
Index statistics are gathered.

Table and distributed statistics
Table and distributed statistics are gathered.

Table and distributed statistics and basic indexes
Table, distributed and basic index statistics are gathered.

Extended statistics for index only
Extended index statistics are gathered.

Extended statistics for indexes and basic table statistics
Extended index statistics and basic table statistics are gathered.

All statistics
All statistics are gathered.

No statistics
None of the statistics are gathered.

This value is the default value.

Unrecoverable load
Specifies whether the transaction for the data load operation unrecoverable. The supported values are `Yes` and `No`. The default value Yes indicates that the data load operation is unrecoverable, meaning that the loaded data cannot be recovered by a subsequent roll forward operation.

Data buffer size
Specifies the number of 4-kilobyte pages in the buffer to use for transferring data within the load utility. The default value 0 specifies that the load utility should automatically determine the optimal value.

Sort buffer size
Specifies the number of 4-kilobyte pages in the buffer to use for sorting index keys in the load utility when the **Indexing mode** property is not set to value `Deferred`. The default value 0 specifies that the load utility should automatically determine the optimal value.

CPU parallelism
Specifies the number of processes or threads that the load utility will create for parsing, converting, and formatting records when building table objects. The default value 0 specifies that the load utility should automatically choose the optimal CPU parallelism value based on the current environment.

Disk parallelism
Specifies the number of processes or threads that the load utility will create for writing data to the table space containers. The default value 0 specifies that the load utility should automatically choose the optimal disk parallelism value based on the current environment.

Copy loaded data
Specifies whether to save a copy of the loaded data.

The supported values are:

No No copy is made. This is the default value.

Yes C copy is made by the user-specified library.

Yes, Use Tivoli® Storage Manager
A copy is made using Tivoli Storage Manager.

Copy to device or directory name
Specifies the name of the device or directory to which to save a copy of the loaded data. There is no default value. This property is applicable when the **Copy loaded data** property is not set to value `No`.

Copy library name
Specifies the name of the shared library containing the backup and restore I/O functions to be used for saving a copy of the loaded data. There is no default value. This property is applicable when the **Copy loaded data** property is set to value **Yes**

Allow access mode
The access level to allow to other applications for the target table to which the data is loaded.

The supported values are:

No access
The LOAD utility locks table for exclusive access during the load.

Read access
The LOAD utility locks the table in shared access mode so that other applications can read data from it while the load is taking place.

Tablespace for read access
Specifies the optional tablespace to use if rebuilding indexes. A shadow copy of the index is built in the specified tablespace and then copied to the original tablespace at the end of the load.

There is no default value.

Set integrity pending cascade
Specifies whether to automatically set Integrity pending state to all the descendent tables.

The supported values are:

Deferred
Only the table that is loaded is placed in the Integrity pending state.

This value is the default value.

Immediate
The Integrity pending state for foreign key constraints is immediately extended to all descended foreign key tables.

Lock with force
Specifies whether the load utility should be allowed to force off other applications that hold conflicting locks on the target table in order to acquire the necessary locks for the load operation. The supported values are `Yes` and `No`. The default value is `Yes`.

Partitioned database configuration
Specifies whether the data is loaded to a table distributed across multiple database partitions. The supported values are `Yes` and `No`.

The default value is `No`.

Output partition numbers
Specifies the comma-separated list of database partitions to which to load the data. The list must be enclosed by parenthesis. The word “to” can be used to specify a range or partitions instead of a single partition, for example: (0, 2 to 10, 15).

There is no default value.

Partitioning partition numbers
Specifies the comma-separated list of database partitions to be used in the distribution process. The list must be enclosed by parenthesis. The word "to" can be used to specify a range or partitions instead of a single partition number, for example: (0, 2 to 10, 15).

There is no default value.

Maximum number of partitioning agents
Specifies the maximum number of partitioning agents in a load session.

The default value is `25`.

Partition errors isolation mode
Specifies the mode for handling errors that occur on individual database partitions.

The supported values are:

Setup errors only
Handle setup errors only.

Load errors only
Handle load errors only.

This value is the default value.

Setup and load errors
Handle setup and load errors.

No isolation
Handle no isolation level errors.

Status interval
Specifies the amount of data to load in megabytes before issuing a progress message.

The valid values are in the range of **1** to **4000**.

The default value is `100`.

Port range
Specifies the range of TCP ports used to create sockets for internal communications. The format is: (lower-port,higher-port).

The default value is `6000,6063`.

Check truncation
Specifies whether to check for data truncations on input/output.

The valid values are `Yes` and `No`.

The default value is `No`.

Trace record number
Specifies the number of records to trace when a review of a dump of the data conversion process and the output of the hashing values is required.

The default value is 0.

Distribution file name
Specifies the name of the database partition distribution file that the LOAD utility should generate. If the value is not specified, the file is not generated.

There is no default value.

Omit distribution map header
Specifies whether to prevent inclusion of the distribution map header in the distribution file.

The supported values are `Yes` and `No`.

The default value is `No`.

Statistics partition number
Specifies the number of the partition for which to collect statistics if the stage was configured to collect statistics. The value -1 specifies to collect statistics on the first database partition in the output database partition list.

The default value is -1.

Loading data to Oracle

These are the properties that can be set to customize the bulk load operation when the stage is configured for the Oracle database type.

Schema name
Specifies the name of the schema (owner) in which the target table resides.

There is no default value.

If no value is specified the table is assumed to reside in the schema of the currently connected user.

Partition name
Specifies the name of the table partition to which to load data.

There is no default value.

If the value is not specified, the data is loaded to the entire table.

Batch size
Specifies the maximum number of input records in a batch.

The default value is `100`.

Load mode
Specifies the method to use to load the data in the target table.

The supported values are:

Automatic
Loads the data directly to the target database table using Direct Path Oracle interface.

Manual
Creates Oracle SQL*Loader control file and data file which can later be used to perform the load with Oracle SQL*Loader utility.

Control file name
Specifies the name of the Oracle SQL*Loader control file that the stage creates when the **Load mode** property is set to value `Manual`.

There is no default value.

Data file name
Specifies the name of the Oracle SQL*Loader data file that the stage creates when the **Load mode** property is set to `Manual`.

There is no default value.

Preserve trailing blanks
Specifies whether to preserve trailing blanks in the input text values or to truncate them.

The supported values are:

Yes Preserve the trailing blanks.

This value is the default value.

No Truncate trailing blanks.

Case sensitive column names
Specifies whether the input link column names should be treated as case sensitive values.

The value that is specified for this property takes precedence over the value specified in **Use quoted identifiers** property under the **Usage** section.

The supported values are:

Yes The column names should be treated as case-sensitive values.

No All characters in the input link column names should be treated as uppercase characters.

This value is the default value.

In the manual load mode the stage does not write data to the database table directly, but instead creates Oracle SQL*Loader control and data files to be used with Oracle SQL*Loader to load the data to the table. The format of these two files is determined internally by the stage and can change in future releases of the product. For this reason it is not recommended to design jobs that rely on the current format of these files.

If the control file name is not specified, the stage will create the control file name consisting of the specified Oracle service name, followed by the underscore character, followed by the specified name of the table to which the data is loaded, followed by the `.ctl` extension.

If the data file name is not specified the stage will use the same logic to produce the data file name except instead of the `.ctl` extension, the `.dat` extension will be used for the file name.

Database lookup operation

You can configure the stage to complete a lookup operation on the database for each input record (referred to as the key record) and return rows that match the criteria that are specified by that record.

The lookup operation is performed by running a parameterized SELECT statement which contains a `WHERE` clause with parameters that are associated with the columns marked as Key columns in the records that represent key records for the lookup.

On server canvas the job configuration for the database lookup requires that the Transformer stage is used in combination with the database stage. The Transformer stage has an input link on which the key records arrive to be used as input for the lookup query. It also has one or more reference links coming from the database stage. The database stage is provided with input key records on this link. For each input key record, the database stage runs the parameterized `SELECT` statement with the key record values used in the `WHERE` clause and provides the corresponding matching records to the Transformer stage. Those records are then processed and

routed by the Transformer stage to one or more of its output links to be further processed by the downstream stages in the job.

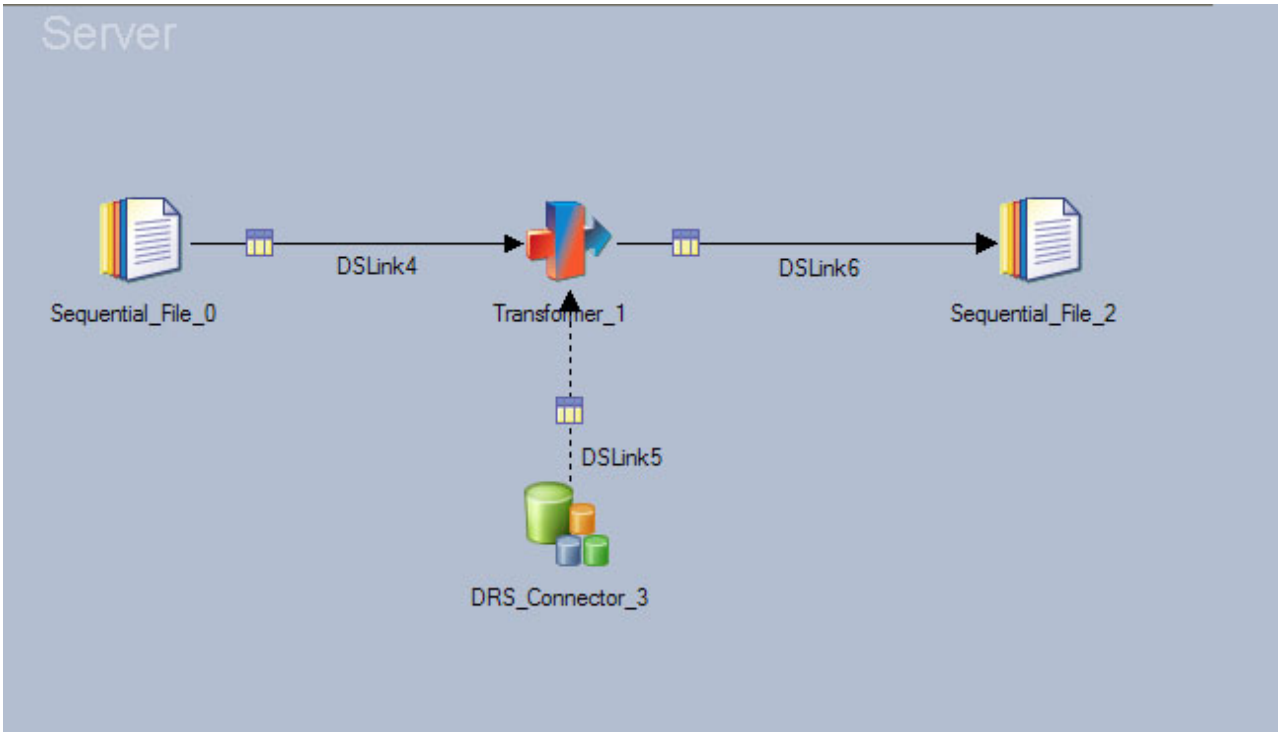
In some cases the SELECT lookup statement may return multiple record matches. The user can specify whether the stage should log a message when this happens.

In the DRS Connector stage this is specified through the **Disable warnings for multiple lookup matches** property. Valid values are:

Yes The warning messages are not logged

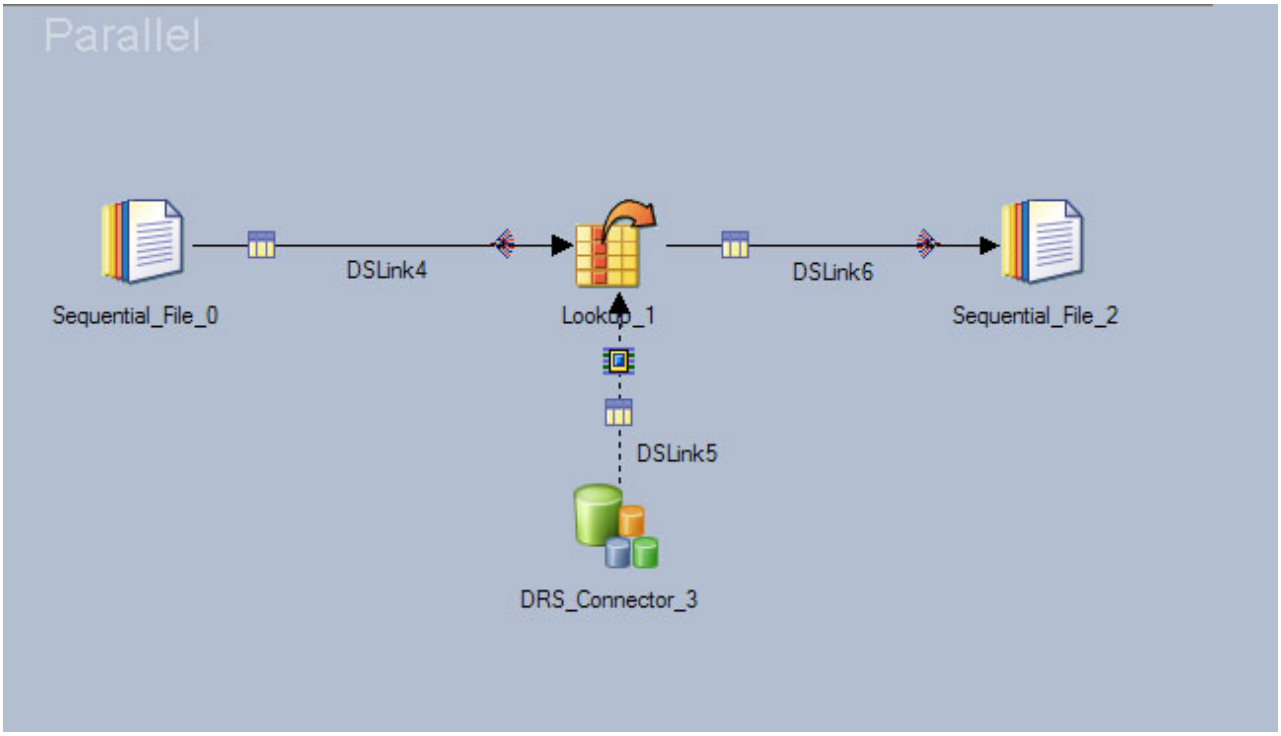
No The warning messages are logged

The following figure shows a DRS Connector stage that is configured for lookup operation in a server job. The records from the input file (**Sequential_File_0**) are passed through the Transformer stage (**Transformer_1**) to the DRS Connector stage (**DRS_Connector_3**) which uses their key column values as input for performing lookup operation on the database. The results of the lookup operation are written to the output file (**Sequential_File_2**).



The configuration for the database lookup on the parallel canvas is similar to one on the server canvas. The Lookup stage is used instead of the Transformer stage.

The following figure shows a DRS Connector stage configured for lookup operation in a parallel job. The records from the input file (**Sequential_File_0**) are used as keys for performing lookup operation on the database data. In case of a Normal lookup, the lookup operation is performed by the Lookup stage (**Lookup_1**). In the case of a Sparse lookup mode, the lookup operation is performed by the DRS Connector stage (**DRS_Connector_3**). The results of the lookup operation are written to the output file (**Sequential_File_2**).



Two lookup modes are supported on parallel canvas and can be specified in the **Lookup type** property on the DRS Connector stage.

Normal lookup (in-memory lookup)

Uses non-parameterized `SELECT` statement in the DRS Connector stage. The stage runs the statement only once for the duration of the job and delivers the results to the Lookup stage. The Lookup stage stores all results in memory. For each key record arriving on its input link the Lookup stage performs the lookup operation on the data in memory to find the matches. The rows that meet the matching criteria are delivered on the output links to the downstream stages. This type of lookup is suitable for small database tables that can fit in system memory. In this case the lookup operation for each input key record is performed in memory and does not required a roundtrip to the database.

Sparse lookup (direct lookup)

Uses a parameterized `SELECT` statement with statement parameters from the WHERE statement clause corresponding to the columns marked as Key columns in the input key records. For each input key record that arrives to the Lookup stage, the parameterized `SELECT` statement is executed to fetch the records from the database that satisfy the `WHERE` clause created for that key record. This type of lookup is suitable for very large database tables where it is not possible or not practical to store the entire database table in memory. To improve the performance of sparse lookup operation it is advisable to define an index on the columns used as key columns (`WHERE` clause parameters) in the parameterized `SELECT` statement used to perform the lookup.

You can use the sparse lookup method only in parallel jobs.

Creating, dropping, and deleting database tables at runtime

You can configure the stage to create, replace or truncate the tables in the database before writing data rows to it.

The following table actions are supported:

Append
The table referenced by the stage must exist in the database.

This is the default table action.

In the DRS Connector stage this table action is specified by setting the **Create table** property under **Table action** property in the **Usage** section to `No`.

Create The stage creates the table in the database.

The `CREATE TABLE` statement is generated automatically by the stage from the specified table name and column definitions on the input link.

In the DRS Connector stage, this table action is specified by setting the **Create table** property under **Table action** property in the **Usage** section to `Yes`, and the **Drop table** property to `No`.

Replace
The stage drops the table (if it exists) from the database and creates the table again.

The `CREATE TABLE` and `DROP TABLE` statements are generated automatically by the stage from the specified table name and column definitions on the input link.

In the DRS Connector stage, this table action is specified by setting **Create table action** and **Drop table** properties in the **Usage** section to `Yes`.

Delete The stage deletes all the rows from the table before writing new rows to the table.

In the DRS Connector stage, this table action is specified by setting the **Delete table** property under **Table action** property in the **Usage** section to either of the following values:

Yes The stage generates and runs the `DELETE TABLE` statement.

Truncate
The stage generates and runs the `TRUNCATE TABLE` statement, when this is supported for the selected database type.

Generating SQL statements in the connector at design time

You can configure the connector to generate SQL statements at design time in their statement properties.

Before you begin

Create a job that includes a connector as a source or target.

About this task

You can generate the SQL statement text only for those statement properties that have the **`Generate SQL statement`** option in the Build list.

Note: Under some circumstances, the connector requires a connection to generate SQL statements. When a user name and password are not supplied and a connection is required, a connection is made by using the user who is running the ASB Agent service.

Procedure

1. Double-click the connector on the job canvas to open the stage editor.
2. In the navigator, click the output or input link, depending on the type of job that you create.
3. Set **Generate SQL at runtime** to No.
4. In the **Table name** property, type the name of the table for the SQL statement.
5. For jobs in target context (input links), select the type of statement you want to generate in the **Write mode** property.
6. On the **Columns** page, define the columns to use in the SQL statement.
7. Click the **Properties** tab.
8. Click the **Build** button that is associated with the statement property, and select **Generate SQL statement** from the list.

   Note: The **Generate SQL statement** option will only be available for statements which that connector supports generating at design time. In some cases a connector may only support generating the SQL at runtime during job execution.
9. Click **OK** to save the job.

Validating SQL statements in the connector at design time

After you generate or write a SQL statement, you can validate the statement during job design.

About this task

You can validate the SQL statement text only for those statement properties that have the **Validate SQL** option in the Build list.

Note: Under some circumstances, the connector requires a connection to validate SQL statements. When a user name and password are not supplied and a connection is required, a connection is made by using the user who is running the ASB Agent service.

Procedure

1. Save the job.
2. Click the **Build** button that is associated with the statement property, and select **Validate SQL**. The **Validate SQL** option is enabled only if the statement property contains a value and this option will only be available for statements which the target RDBMS supports validating.

Results

The connector validates the SQL statement by preparing the statement with the RDBMS it supports. If the SQL contains error, an error message is shown.

Data type support

Different databases support proprietary data types, which the stage needs to convert to and from the InfoSphere DataStage data types that are specified for the column on the stage links.

For example, when the stage is configured to create a table in the target database, it constructs the `CREATE TABLE` statement and for each column on the link it specifies the data type to be used in the target table. For some InfoSphere DataStage data types this mapping is supported and for some it is not.

When the job runs and the stage reads data from the database and writes data to the database, it needs to convert data between the column data types in the database and the InfoSphere DataStage data types for the corresponding columns on the link. The following topics provide mapping details for each of the supported database types.

The topics include conversion tables that show how the stages map InfoSphere DataStage data types to target database types when constructing the `CREATE TABLE` statement.

The tables should be used as a guide when choosing which InfoSphere DataStage data type to use for columns on the link depending on the column definitions in the database table. For a particular database type check if that data type appears in the last column in the conversion table. If the database type is not listed in the conversion table, choose the InfoSphere DataStage type that is the closest possible match for that database data type. Ensure that the *Length*, *Scale*, *Extended*, and *Nullable* attributes for the column on the link are set to values that match the column definition in the database.

When starting with an empty link and when the selection of column types is not dictated by the other stage on the link, the preferred option for specifying columns on the link is to use the metadata import wizards available in InfoSphere DataStage and QualityStage Designer. The wizards allow you to import target table definition into the metadata repository. Once imported, the table definition can be applied to the links of the stage. This can be done by clicking the Load button on the Columns tab in the stage dialog, or the table definition can be dragged and dropped from the Repository view directly onto the link.

For importing table definitions to be used with the DRS Connector stage the preferred wizard to use is the Connector Import Wizard.

For the link columns of `Decimal` and `Numeric` data types it is always necessary to specify the Length attribute. This attribute represents the decimal precision. The *Scale* attribute represents the decimal scale and is optional. When the *Scale* attribute is not specified then the scale of zero is assumed.

For the DRS Connector stage the *Length* can be omitted in certain cases as explained in the following topics, but it is recommended to provide the *Length* value.

IBM DB2 data type support

When a job runs, the stage maps InfoSphere DataStage data types to IBM DB2 data types.

Table 8. Mapping of InfoSphere DataStage data types to IBM DB2 data types

SQL type	Length	Scale	Extended	IBM DB2 Column Definition
BigInt	n/a	n/a	any	BIGINT
Binary	*n*	n/a	n/a	CHAR(*n*) FOR BIT DATA[1]
Bit	n/a	n/a	n/a	SMALLINT
Char	*n*	n/a	blank	CHAR(*n*)[1]
Char	*n*	n/a	Unicode	GRAPHIC(*n*)[1]
Date	n/a	n/a	n/a	DATE
Decimal	*p*	*s*	n/a	DECIMAL(*p,s)*
Double	*p*	n/a	n/a	DOUBLE
Float	*p*	n/a	n/a	FLOAT
Integer	n/a	n/a	any	INTEGER
LongNVarChar	*n*	n/a	n/a	DBCLOB
LongVarBinary	*n*	n/a	n/a	BLOB
LongVarChar	*n*	n/a	blank	CLOB
LongVarChar	*n*	n/a	Unicode	DBCLOB
NChar	*n*	n/a	n/a	NCHAR(*n*)
NVarChar	*n*	n/a	n/a	VARGRAPHIC(*n*)[1]
Numeric	*p*	*s*	n/a	DECIMAL(*p,s*)
Real	n/a	n/a	n/a	REAL
SmallInt	n/a	n/a	any	SMALLINT
Time	n/a	*s*	any	TIME
Timestamp	n/a	*s*	any	TIMESTAMP
TinyInt	n/a	n/a	any	SMALLINT
VarBinary	*n*	n/a	n/a	VARCHAR(n) FOR BIT DATA[1]
VarChar	*n*	n/a	blank	VARCHAR(*n*)[1]
VarChar	*n*	n/a	Unicode	VARGRAPHIC(*n*)[1]

Table notes:

1. If the Length is left blank the DRS Connector stage assumes the length of 32.

ODBC data type support

When a job runs, the stage maps InfoSphere DataStage data types to target database types.

Multiple data types can be listed in the table because the actual target data type depends on the ODBC driver in question. In some instances, the ODBC driver will not accept the target data type specified by the stage when creating the table. For example, the Dynamic RDBMS stage will map the*Binary* source data type to the *CHAR(n) FOR BIT DATA* target data type, which will be accepted by DB2 driver but not by Oracle.

The database can accept the data type specified by the stage, but will then internally convert it to a more suitable native data type.

The generic data type is the type that the stage specifies in the auto-generated `CREATE TABLE` statement. The mapping from this data type to the actual database type in the backend database is performed by the ODBC driver. If the mapping is not supported, the `CREATE TABLE` statement fails.

Table 9. Mapping of InfoSphere DataStage data types to ODBC data types

SQL type	Length	Scale	Extended	Generic Column Definition
BigInt	n/a	n/a	any	BIGINT, NUMBER
Binary	*n*	n/a	n/a	Not supported
Bit	n/a	n/a	n/a	SMALLINT
Char	*n*	n/a	blank	CHAR(*n*)
Char	*n*	n/a	Unicode	CHAR(*n*)
Date	n/a	n/a	n/a	DATE, TIMESTAMP, DATETIME
Decimal	*p*	*s*	n/a	DECIMAL(*p*,*s*)
Double	*p*	n/a	n/a	DECFLOAT, NUMBER, FLOAT
Float	*p*	n/a	n/a	REAL, NUMBER
Integer	n/a	n/a	n/a	INTEGER, INT, NUMBER
LongNVarChar	*n*	n/a	n/a	VARCHAR(n), VARCHAR2(n) NTEXT, LONG
LongVarBinary	*n*	n/a	n/a	Not supported
LongVarChar	*n*	n/a	blank	VARCHAR, VARCHAR2
LongVarChar	*n*	n/a	Unicode	VARCHAR, VARCHAR2
NChar	*n*	n/a	n/a	CHAR(*n*)
NVarChar	*n*	n/a	n/a	VARCHAR(*n*)
Numeric	*p*	*s*	n/a	DECIMAL(*p*,*s*)
Real	n/a	n/a	n/a	REAL, NUMBER
SmallInt	n/a	n/a	any	SMALLINT, NUMBER
Time	n/a	any	any	TIME, TIMESTAMP, DATETIME
Timestamp	n/a	any	any	TIMESTAMP, DATETIME
TinyInt	n/a	n/a	any	TINYINT, SMALLINT,NUMBER
VarBinary	*n*	n/a	n/a	Not supported
VarChar	*n*	n/a	blank	VARCHAR(*n*), VARCHAR2(*n*)
VarChar	*n*	n/a	Unicode	VARCHAR(*n*), VARCHAR2(*n*)

Oracle data type support

When a job runs, the stage maps InfoSphere DataStage data types to Oracle data types.

The Oracle DATE data type supports year, month, day, hour, minute and second portions. On the other hand InfoSphere DataStage Date data type supports only year, month and day portions. Oracle *TIMESTAMP* and InfoSphere DataStage *Timestamp* data types support year, month, day, hour, minute, second, and fractional seconds.

Oracle does not allow empty string values for columns of character Oracle data types of variable size (such as *VARCHAR2*, *NVARCHAR2*, *CLOB*, and *NCLOB*).

InfoSphere DataStage supports empty string values. When an empty string value is provided to Oracle, it writes it to the database as a NULL value. This is important to note when considering the nullable attribute of the columns on the link. For example, an empty string value is valid for a link column for which Nullable attribute is set to No. However, it is invalid for an Oracle column that does not allow NULL values (marked as NOT NULL).

Similar consideration needs to be made when writing values to Oracle columns of character data types of fixed size (such as *CHAR* and *NCHAR)*. When an empty string value is provided to Oracle to write to these columns, Oracle will not pad the value with spaces; Oracle will write NULL value to the column. If the table column does not allow NULL values that will result in error.

Table 10. Mapping of InfoSphere DataStage data types to Oracle data types

SQL type	Length	Scale	Extended	Oracle Column Definition
BigInt	n/a	n/a	any	NUMBER(19)
Binary	n	n/a	n/a	RAW(n)[1]
Bit	n/a	n/a	n/a	NUMBER(5)
Char	*n*	n/a	blank	CHAR(n)[1]
Char	*n*	n/a	Unicode	NCHAR(n)[2]
Date[3]	n/a	n/a	n/a	DATE
Decimal	*p*	*s*	n/a	NUMBER(*p*,*s*)
Double	*p*	n/a	n/a	BINARY_DOUBLE
Float	*p*	n/a	n/a	BINARY_FLOAT
Integer	n/a	n/a	any	NUMBER(10)
LongNVarChar	*n*	n/a	n/a	NCLOB
LongVarBinary	*n*	n/a	n/a	BLOB
LongVarChar	*n*	n/a	blank	CLOB
LongVarChar	*n*	n/a	Unicode	NCLOB
NChar	n/a	any	n/a	NCHAR(1000)
NChar	*n*	n/a	n/a	NCHAR(*n*)[2]
NVarChar	*n*	n/a	n/a	NVARCHAR2(*n*)[1]
Numeric	*p*	*s*	n/a	NUMBER(*p*,*s*)
Real	n/a	n/a	n/a	BINARY_FLOAT
SmallInt	n/a	n/a	any	NUMBER(5)
Time[4]	n/a	blank	blank	DATE
Time[4]	n/a	s	blank	TIMESTAMP(*s*)
Time[4]	n/a	blank	Microseconds	TIMESTAMP(6)
Time[4]	n/a	s	Microseconds	TIMESTAMP(*s*)
Timestamp	n/a	blank	blank	DATE
Timestamp	n/a	*s*	blank	TIMESTAMP(*s*)
Timestamp	n/a	blank	Microseconds	TIMESTAMP(6)
Timestamp	n/a	*s*	Microseconds	TIMESTAMP(*s*)
TinyInt	n/a	n/a	any	NUMBER(3)
VarBinary	*n*	n/a	n/a	RAW(*n*)[1]

Table 10. Mapping of InfoSphere DataStage data types to Oracle data types (continued)

SQL type	Length	Scale	Extended	Oracle Column Definition
VarChar	*n*	n/a	blank	VARCHAR2(n)[5]
VarChar	*n*	n/a	Unicode	NVARCHAR2(n)[1]

Table notes:

1. If the Length is left blank, the DRS Connector stage assumes the length of 2000.
2. If the Length is left blank, the DRS Connector stage assumes the length of 1000.
3. When a column of InfoSphere DataStage Date type is used on the input link to write to an Oracle DATE or TIMESTAMP column, the hour, minute and second portions in the target value are set to midnight time. To change this default behavior you must use the *CC_ORA_DEFAULT_DATETIME_TIME* environment variable. Its format is HH:MI:SS where HH represents hours in 24-hour notation, MI represents minutes and SS represents seconds. When the environment variable is set, the stage uses the specified value for the default hour, minute and second portion for the target values. The environment variable does not provide an option to specify default fractional seconds when writing to Oracle TIMESTAMP. To be able to control fractional seconds you must use InfoSphere DataStage Time or Timestamp column on the link.
4. When a column of InfoSphere DataStage Time type is used on the input link to write to an Oracle DATE or TIMESTAMP column, the month, day and hour portions in the target value are set to current date (the exception is DRS Connector stage when the write mode is not Bulk load in which case zeros are used as default). To change this default behavior you must use the *CC_ORA_DEFAULT_DATETIME_DATE* environment variable. Its format is YYYY-MM-DD where YYYY represents years, MM represents months and DD represents days. When the environment variable is set, the stage uses the specified value for the default year, month and day portion for the target values.
5. If the Length is left blank, the DRS Connector stage assumes the length of 4000.

Considerations for processing large object (LOB) values

The stage can read and write large object (LOB) data types in the backend databases. The maximum size of the LOB values that are exchanged between the stage and the database needs to be carefully considered and the job design needs to be planned accordingly.

On the InfoSphere DataStage Server jobs the maximum size of the records on the link is limited by the system resources.

The Length attribute for any column on the link should be set to the smallest value large enough to accommodate the actual values that will be represented by that column when the job runs.

Note: In InfoSphere DataStage server jobs it is possible to use job parameter to dynamically specify the Length for the columns on the stage links. To do this, the *Description* attribute of the column needs to be set to a value in the following format:

```
param{length=#job_parameter_name#}
```

where *job_parameter_name* is the name of the job parameter through which the column length is specified when the job runs.

On the InfoSphere DataStage Parallel canvas in addition to the constraints dictated by the available system resources, the maximum size of records is also limited by the *APT_DEFAULT_TRANSPORT_BLOCK_SIZE* environment variable. This environment variable is a predefined InfoSphere DataStage environment variable

that is automatically defined at the InfoSphere DataStage project level. The default value is 65536 bytes. To transfer records of a larger size, this variable needs to be increased accordingly.

For example, assume that the stage is configured to read data from a database table that contains one integer column and two LOB columns. And that the maximum size of the values in each of the LOB columns is 1 MB. The *APT_DEFAULT_TRANSPORT_BLOCK_SIZE* would then typically be set to 2097152 (2 MB) to accommodate for the values in the two LOB columns, plus another 65536 bytes to accommodate for the integer column and the internal record data in each record. So the actual value to specify would be 2162688, which could then further be rounded to the value of 2200000.

Transaction control

Dynamic relational database stages provide two levels of control for database transactions.

Size Specifies the number of rows per transaction

Isolation level
Specifies the type of transactions.

Transaction size

The transaction size is specified as the number of records to write in a transaction before committing the transaction and beginning the new one.

The transaction size can be specified when the stage is writing rows to the database except when it is writing the rows in bulk load mode.

If the number of rows in the last transaction in the job is smaller than the specified transaction size, the transaction is still committed.

Each time an array of rows is sent to the database, the stage checks if the number of rows that were written in the current transaction reached the specified transaction size. If the specified transaction size was reached, the stage commits the transaction and starts a new one. When writing rows to the database in array mode, the *Transaction size* should be set to a value that is a multiple of the specified *Array size*.

The transaction size of zero means that all rows that the stage receives on the input link for the duration of the job are written to the database in a single transaction.

In the DRS Connector stage the transaction size is specified in the **Record count** property in the **Transaction** category in the **Usage** section. The allowed values are between 0 and 999,999,999. The default value is 2,000, which matches the default value for **Array size** property for this stage.

Transaction isolation level

The transaction isolation level can be specified for the stage reading rows from the database or writing rows to the database. It specifies the type of transaction in terms of the locks that are acquired on the database objects affected by the transaction and that are kept on behalf of the stage for the duration of the transaction.

The dynamic relational stages provide a generic list of transaction isolation levels which are internally mapped to the corresponding isolation level supported by the backend database systems.

The supported transaction isolation levels in dynamic relational stages are:

Read Uncommitted
The transaction can see the changes made by other transactions that have not yet been committed. These are referred to as dirty reads.

Read Committed
The transaction cannot see the changes made by other transactions that have not yet been committed. Therefore dirty reads are not possible.

If the same statement is executed for the second time in the same transaction it can see the changes made to the rows that have been committed by other transaction after the current transaction started.

If the same statement is executed twice in the same transaction, it can retrieve rows with different values. These are referred to as non-repeatable reads.

This isolation mode also does not prevent the phantom reads. Phantom reads happen when the same statement is executed twice and the second statement fetches additional rows that have been inserted and committed by another transaction.

Repeatable Read
Similar to the Read Committed isolation level except that the changes committed by other transactions are not seen by the statements performed in the current transaction. This means that the non-repeatable reads are not possible.

Phantom reads are possible.

Serializable
Similar to the Repeatable Read isolation level except that phantom reads are prevented.

This is the most restrictive isolation level in terms of locks acquired on the database objects accessed by the transaction. It enforces the consistency within the transactions at the expense of their concurrency.

Not all databases support these same four levels of transaction isolation and in some cases they use different terminology for isolation levels. The dynamic relational stages map the value for the isolation level specified by the user to the closest supported value defined for the database type for which the stage was configured.

In the DRS Connector stage the transaction isolation level is specified in the **Isolation level** property in the **Transaction** category in the **Usage** section. The default value is `Read committed`.

Record ordering

You can control the order in which the stage processes records that arrive on multiple input links that are defined for the stage.

This feature is available only in parallel jobs.

Before setting the properties that control the order of records on the input links of the stage, it is necessary to order the input links. The logical order of the links is configured in the **Link Ordering** page of the stage dialog.

Note: The visual representation of the links for the stage that is displayed on the job canvas does not necessarily match the logical order of the links in which the links are processed at runtime.

The **Record ordering** property defines the order in which the input records are processed.

Valid values are:

All records
all available records on the first links are processed, followed by all available records on the second link, and so forth until all the input links are processed.

The links are processed in the order specified on the **Link Ordering** page.

First record
The records are processed in a round-robin fashion. The first record from the first link is processed, followed by the first record on the second link, and so forth. Once the first records on all of the links are processed, the second record on the first link is processed, followed by the second record on the second link, and so forth. This pattern continues until all the records on all the links are processed.

Ordered
The ordering of the records is defined by the by the dedicated link columns (key columns) values in the records, and by the remaining properties under the **Record ordering** property.

The records on each link are sorted based on the values that they have for the key columns. Multiple columns can be assigned as key columns, and they are referred to as **Key column [n]**.

The **Key column [n]** property is defined for each key column in the record. The records are sorted based on their values for **Key column [1]** , and then the records that have the same value for this column are sorted based on their values for each additional **Key column [n]** , until all the key columns are applied.

By default only Key column [1] property is available.

To add property for another key column, right click on any existing **Key column [n]** property and select option **Add Property Value**.

To remove a **Key column [n]** property, right click on it and select option Remove Property Value. Each Key column [n] property contains the following properties:

Column name
Specifies the name of the key column on the link. The column with this name needs to have the same definition on all input links.

The name can be typed in or selected by clicking on the **Available columns** button.

Sort order
Specifies the order in which the records are sorted for the key column.

The order can be set to `Ascending` or `Descending`.

Null order
Specifies how `NULL` values for the key column are sorted.

Valid values are:

Before The `NULL` values are appear before the non-`NULL` values.

After The `NULL` values are appear after the non-`NULL` values.

Case sensitive
Specifies whether the values are sorted in case-sensitive order or case-insensitive order for the key column. Valid values are:

Yes The sorting is case-insensitive.

No The sorting is not case-insensitive.

Before SQL and After SQL statements

You can run custom SQL statements on the target database before and after data is processed.

Before SQL
Custom SQL statements are run when the job starts and before any data is processed by the stage.

After SQL
Custom SQL statements are run after all the data has been processed by the stage and just before ending the job.

If multiple statements are specified as Before SQL or After SQL statements, they need to be separated by a semicolon character.

The stage runs Before SQL statements in a one database transaction and After SQL statements in another. If a pair of semicolons is encountered in the statements, the stage interprets it as the user-defined commit point. It commits the current transaction and starts a new one.

To configure the DRS Connector stage to run Before SQL and After SQL statements you must set the **Before/After SQL** category in **Usage** section to `Yes`. The Before SQL and After SQL statements are then specified in the **Before SQL** and **After SQL** properties located under the **Before/After SQL** category.

You can specify Before SQL and After SQL statements in a file located on the InfoSphere DataStage Server machine. To point the stage to this file, instead of typing in the actual statements in the stage you must type `FILE=` *`filepath`* or `{FILE}` *`filepath`*, where *filepath* is the fully-qualified path to the file with the actual SQL statements.

The stage rolls back the current transaction and fails the job when any statement in the Before SQL and After SQL lists of statements fails to run successfully. You can change this default behavior.

To cause the DRS Connector stage to continue processing statements when this happens, you must set the properties **Continue if Before SQL failed** and **Continue if After SQL failed** under the **Before/After SQL** category in the **Usage** section to **Yes**.

Data errors

Data errors occur when a stage fails to write a particular row to the database.

When a DRS Connector stage fails to write a particular row to the database the behavior depends on whether the job in question is a server job or a parallel job. In case of a server job, it reports a warning and continues to process the remaining rows. In case of a parallel job, it immediately reports the error and stops the job.

You can change the behavior of the DRS Connector stage through the **Fail on row error** property located under **Session** category in the **Usage** section. When set to `Yes`, the stage reports an error for any row that it could not write to the database. When set to `No`, the DRS Connector stage logs a warning and continues to process rows.

The *CC_MSG_LEVEL* environment variable is used to control the level of information that the DRS Connector stage writes to the job log. The *CC_MSG_LEVEL* environment variable values and their options are:

1 Trace, Debug, Informational, Warning and Fatal errors are logged.

Includes Trace level messages, which are the lowest level messages. Note that specifying this value can result in very large job log.

2 Debug, Informational, Warning and Fatal errors are logged.

Typically used for troubleshooting problems in the DRS Connector because it results in logging debug level messages which are otherwise not logged by default.

3 Informational, Warning and Fatal errors are logged.

This is the default value.

4 Warning and Fatal errors are logged.

5 Fatal errors are logged.

Rejecting bad records

For a server job, you can configure the stage to reject bad records.

For this functionality to work the stage must not be processing records in array mode. The Array size property must be set to 1.

To reject rows that arrive on an input link of the stage and that could not be written to the database, the link must be coming from a Transformer stage. In the Transformer stage there must be another output link defined and marked as reject link and configured to accept records that could not be processed by the database stage. The records are sent to the reject link and are delivered to the downstream stage which then processes or stores them. For example, a Sequential File stage can be used to store the rejected records to a file for later analysis.

Note: The reject links in Transformer stage can be configured to accept records that could not be processed by the database stage on another link, or that could not be written to another link due to a constraint defined for that link in the Transformer stage.

Handling $ and # characters

InfoSphere DataStage does not allow for the use of # and $ characters in the column names on the links.

These restrictions may not exist for the column names in the database tables. InfoSphere DataStage supports an environment variable that can be used to handle this situation. The name of the environment variable is *DS_ENABLE_RESERVED_CHAR_CONVERT*, a predefined environment variable in InfoSphere DataStage projects. Its value can be modified at the InfoSphere DataStage project level, or it can be added to a particular job as job parameter and set at the job level. It can be set to `Yes` or `No`. The default value is `Yes`.

The value `Yes` replaces `__035__` sequence of characters in the specified table and column names with # character. It also replaces `__036__` sequence of characters with a single $ character.

When you use the stage in a server job, specify the database column names in the user-defined SQL statements in their original form, including any # and $ characters in them. The column names on the links have the restriction for using # and $ characters. They are represented in the SQL statement text by question marks (bind parameter markers). The question marks are bound to the columns on the link by position in a top-down fashion, so that way the names of the columns on the link do not need to be present in the SQL statement text.

For example, for an `UPDATE` statement the following text can be entered:

```
UPDATE table_name SET ##B$ = ? WHERE $A# = ?
```

The association of question marks with column names on the link is not affected by the names of the columns on the link. Instead the first column on the link marked as Key column is associated with the question mark bind parameter in the `WHERE` clause and the first column on the link not marked as Key column is associated with the question mark bind parameter in the SET clause.

In the Dynamic RDBMS stage, the *DS_ENABLE_RESERVED_CHAR_CONVERT* environment variable is supported by the following flavors:

- Informix
- Informix Bulk Load
- MSSQL Bulk Load
- Oracle OCI Bulk Load
- Sybase
- Sybase Bulk Load
- DB2 Bulk Load

The following flavors behave as if the *DS_ENABLE_RESERVED_CHAR_CONVERT* environment variable was set by default and automatically makes the conversion:

- Oracle OCI
- ODBC
- DB2

SQL meta tags

SQL meta tags are platform-independent SQL functions that are supported by the Dynamic Relational Database stages.

At run time, these tags are translated into native database-specific SQL functions of the backend database. They can be applied wherever you provide SQL statements:

- User-defined SQL statements for writing data to the database
- User-defined `SELECT` statement for reading data from the database
- Before SQL and After SQL statements
- `WHERE` and other clauses for automatically generated `SELECT` statement

The use of SQL meta tags is supported also for the SQL statements in a file to which the stage points.

The following list describes the available SQL meta tags and their usage:

%Abs(x)
Returns the absolute value of the numeric argument.

%Coalesce(expr1, expr2, ...)
Returns the first non-null argument provided to the function. The arguments are expressions of any type.

%Concat
Used as an operator for concatenating strings. At runtime this meta tag is replaced by the string concatenation mechanism appropriate for the specific relational database. For example, in IBM DB2 it is replaced with `CONCAT` function that for arguments takes the operands specified on the left and right sides of the %Concat meta tag, while in Sybase it is replaced with a "+" operator. This meta tag is supported with the same limitations as the native concatenation mechanism on the specific relational database. For example, some databases allow concatenating a string with a numeric value while other databases flag this as an error. The Dynamic Relational Database stages make no attempt to check or convert the data types of either of the operands.

%CurrentDateIn
Returns the current date and can be referenced in the user-defined `INSERT` statement or the `WHERE` clause of the user-defined `SELECT`, `UPDATE` or `DELETE` statement.

%CurrentDateTimeIn
Returns the current datetime (timestamp) and can be referenced in the user-defined `INSERT` statement or the `WHERE` clause of the user-defined `SELECT`, `UPDATE` or `DELETE` statement.

%CurrentDateTimeOut
Returns the current datetime (timestamp) in string format and can be referenced in the select list of the user-defined `SELECT` statement.

%CurrentTimeIn
Returns the current time and can be referenced in the user-defined `INSERT` statement or the `WHERE` clause of the user-defined `SELECT`, `UPDATE` or `DELETE` statement.

%CurrentTimeOut
Returns the current time in string format and can be referenced in the select list of the user-defined `SELECT` statement.

%DateAdd(date_from, add_days)
Adds the number of days specified in the **add_days** argument to the date specified in **date_from** argument and returns the result. The **add_days** argument can be a negative number.

%DateDiff(date_from, date_to)
Returns the difference in number of days between two dates (**date_from** and **date_to**) passed as arguments.

%DateIn(dt)
Converts the date value to a format suitable for use in conditional expressions of the WHERE clause in the user-specified SELECT, UPDATE or DELETE statement. or for passing date values in the INSERT statement parameters. The dt argument is a date value or a date string literal in the form YYYY-MM-DD.

%DateOut(dt)
Converts the date value **dt** to a string format suitable for use in the select list of the user-specified SELECT statement.

%DatePart(DTTM_Column)
Returns the date portion of the specified datetime column.

%DateTimeDiff(datetime_from,datetime_to)
Returns a time value representing the difference between two datetimes in minutes.

%DateTimeIn(dtt)
Converts the datetime (timestamp) value to a format suitable for use in conditional expressions of the WHERE clause in the user-specified SELECT, UPDATE or DELETE statement or for passing datetime values in the INSERT statement parameters. The **dtt** argument is a timestamp value or a timestamp string literal in the form YYYY-MM-DD-hh:mm:ss.ffffff.

%DateTimeOut(datetime_col)
Converts the datetime (timestamp) value to a string format suitable for use in the select list of the user-specified SELECT statement.

%DecDiv(a,b)
Returns a floating point number representing the value of **a** divided by **b** , where **a** and **b** are numeric expressions.

%DecMult(a,b)
Returns a floating pointer number representing the value of **a** multiplied by **b** , where **a** and **b** are numeric expressions.

%DTTM(date,time)
Combines the specified date value with the specified time value and returns the resulting datetime (timestamp) value.

%FullJoin(TableName1OrJoin,[tableAlias1],tableName2OrJoin,[tableAlias2], joinCondition)
Produces a full outer join expression for the specified tables.

TableName1OrJoin is the name of the first table or nested join in the join.

tableAlias1 is an alias for the first table in the join and is optional.

tableName2OrJoin is the name of the second table or nested join in the join.

tableAlias2 is an alias for the second table in the join and is optional.

joinCondition is the expression describing the join condition.

%LeftJoin(TableName1OrJoin,[tableAlias1],tableName2OrJoin,[tableAlias2], joinCondition)

Produces a left outer join expression for the specified tables.

TableName1OrJoin is the name of the first able or nested join in the join.

tableAlias1 is an alias for the first table in the join and is optional.

tableName2OrJoin is the name of the second table or nested join in the join.

tableAlias2 is an alias for the second table in the join and is optional.

joinCondition is the expression describing the join condition.

%Like("literal")

Returns the LIKE expression that can be used as condition in `SELECT` statements to look for values that contain the specified literal string value. It generates the following expression:

like 'literal%'

If the literal value contains '_' or '\%' the %Like meta tag resolves to the following:

like 'literal%' escape '\'

%LikeExact(fieldname,"literal")

Returns the expression that can be used as condition in `SELECT` statement to look for the specified literal values. It generates the following expression:

= 'literal'

If the literal ends with the '%' wildcard the generated expression is:

like 'literal' [escape '\']

%NumToChar(Number)

Converts the specified numeric value into a character value.

%RightJoin(TableName1OrJoin,[tableAlias1],tableName2OrJoin,[tableAlias2], joinCondition)

Produces a right outer join expression for the specified tables.

TableName1OrJoin is the name of the first able or nested join in the join.

tableAlias1 is an alias for the first table in the join and is optional.

tableName2OrJoin is the name of the second table or nested join in the join.

tableAlias2 is an alias for the second table in the join and is optional.

joinCondition is the expression describing the join condition.

%Round(expression,factor)

Rounds the specified numeric expression to the specified scale factor before or after the decimal point. If factor is a string literal, it can be rounded to a negative number.

%Substring(source_str,start,length)

Returns the substring of the specified source string at the specified position and length.

%TimeAdd(datetime,add-minutes)

Adds the specified number of minutes (can be negative) to the specified datetime (timestamp) value.

%TimeIn(tm)

Converts the time value to a format suitable for use in conditional expressions of the `WHERE` clause in the user-specified SELECT, UPDATE or

DELETE statement or for passing date values in the INSERT statement parameters. The tm argument is a time value or a date string literal in the form hh:mm:ss.ffffff.

%TimeOut(time_col)
Converts the time value to a string format suitable for use in the select list of the user-specified SELECT statement.

%TimePart(DTTM_Column)
Returns the time portion of the specified datetime column.

%TrimSubstr(source_str,start,length)
Returns the substring of the specified source string at the specified position and length. Any trailing whitespace characters are trimmed from the result.

%Upper(charstring)
Converts the specified string to uppercase. Wildcard characters (like %) can be present in the argument.

Table aliases and table joins

Aliases for the tables can be specified as the second and fourth argument in the SQL for outer, right and left join SQL meta tags.

For example:

```
SELECT Table1Alias.x, Table1Alias.y
FROM %LeftJoin (Table1, Table1Alias, Table2, Table2Alias, Table1Alias.x = Table2Alias.x)
WHERE Table1Alias.y > 2
```

will typically be converted by the stage to following native SQL:

```
SELECT Table1Alias.x,Table1Alias.y
FROM Table1 Table1Alias LEFT OUTER JOIN Table2 Table2Alias
ON Table1Alias.x = Table2Alias.x
WHERE Table1Alias.y > 2
```

Nested outer joins

SQL meta tags for joins can be used recursively to produce nested outer joins.

For example:

```
SELECT Table1.x
FROM %LeftJoin (%LeftJoin (Table1,, Table2,, Table1.x=Table2.x),,
 Table3,, Table1.z = Table3.z AND Table2.q = Table3.q)
WHERE Table1.a = `xyz'
```

will typically be converted by the stage to following native SQL:

```
SELECT Table1.x
FROM Table1 LEFT OUTER JOIN Table2 ON TABLE1.x = Table2.x
LEFT OUTER JOIN Table3 on Table1.z = Table3.z AND Table2.q = Table3.q
WHERE Table1.a = `xyz'
```

Migration from the Dynamic RDBMS stage to the DRS Connector stage

Jobs with Dynamic RDBMS stages can be automatically migrated to the jobs that use DRS Connector stages.

The migration process is performed using the tool IBM InfoSphere Connector Migration Tool, which is available on the client tier machine of IBM InfoSphere

Information Server. The tool is accessible from the Start menu, under **All programs** > **IBM InfoSphere Information Server** > **IBM InfoSphere Connector Migration Tool**.

Once the tool is started and connection is established to the InfoSphere DataStage project, the jobs with DRS plug-in stages are detected and can be selected for migration. Before performing the migration customize the migration preferences through the Preferences... main menu option. For example if migrating jobs with DRS plug-in stages that are connecting to Oracle databases, select the variant that corresponds to the Oracle client version installed on the IBM InfoSphere DataStage Server machine (engine tier). This selection is available under the **Variant selection** tab in the **Preferences** dialog.

The following sections provide additional notes and troubleshooting information that is applicable to specific migration aspects and scenarios.

Migration options for Informix, Sybase and MS SQL Server database types

The DRS Connector stage does not support Informix, Sybase and Microsoft SQL Server database types.

When a Dynamic RDBMS stage configured for one of these database types is migrated to a DRS Connector stage, the **Database type** property in the DRS Connector stage is set to value `ODBC`.

In order for the migrated job to run, you must ensure that the ODBC data source definition (DSN) specified for the **Connection name** in the DRS Connector stage is defined on the system and configured to point to the same database to which the original Dynamic RDBMS stage was pointing. InfoSphere Information Server provides a set of ODBC drivers. Drivers for Informix, Sybase and MS SQL Server database types are also included.

If the Update action property in the original Dynamic RDBMS stage was set to `Bulk insert`, the **Write mode** in the target DRS Connector stage will be set to value `Insert`. The stage that originally used to write data to the database through the native bulk load interface provided by the backend database will be inserting the data using SQL INSERT statements instead.

Note: The Sybase and Microsoft SQL Server ODBC drivers included with Information Server support bulk write mode of operation. This option can be configured for the ODBC DSN definitions used by the DRS Connector stages that were migrated from the Dynamic RDBMS stages configured for Microsoft SQL Server or Sybase bulk load write mode.

Schema reconciliation messages in the job log

When a job with a Dynamic RDBMS stage is converted to the job with a DRS Connector stage, the new job can report schema reconciliation messages (of warning or informational severity) which have not been reported in the original job.

This will typically be the case for DRS Connector stages configured for the IBM DB2 or ODBC database types. For the DRS Connector stages configured for Oracle database type, the schema reconciliation messages will less likely be reported because in this case the DRS Connector will defer most of the schema reconciliation logic to the Oracle database.

A DRS Connector stage configured for DB2 or ODBC database type will report schema reconciliation messages if it determines that the definition of the columns on the link does not match the definition of the corresponding columns in the database table.

There are generally three approaches for eliminating the schema reconciliation messages:

- Modify column definitions on the link to match (as close as possible) the corresponding column definitions in the database table.
- Modify column definitions in the database table to match (as close as possible) the corresponding column definitions on the link.
- Define a message handler that demotes or suppresses the schema reconciliation messages. The message handler definition can be applied to all the jobs in the project or to an individual job.

Example 1

For example, if a parallel InfoSphere DataStage job contains a DRS Connector stage configured for IBM DB2 database type and the stage is used to insert data to a DB2 database table, the input link of the stage contains the following columns:

Table 11. Example 1: Input link contains the following columns

Name	Type	Length	Scale	Extended
C1	Decimal	8	4	(not set)
C2	NVarChar	10	(not set)	(not set)
C3	VarChar	10	(not set)	(not set)
C4	Float	(not set)	(not set)	(not set)
C5	Timestamp	(not set)	(not set)	Microseconds

The target table in the database contains the following columns:

Table 12. Example 1: Target table contains the following columns

Name	Type	Length	Scale
C1	DECIMAL	6	2
C2	VARCHAR	10	(n/a)
C3	VARCHAR	5	(n/a)
C4	INTEGER	(n/a)	(n/a)
C5	TIMESTAMP	(n/a)	3

The job runs and the following messages are reported in the job log:

```
#1
Type: Warning
Message Id: IIS-CONN-DAAPI-00398
Message: DRS_Connector_1: Schema reconciliation detected a
size mismatch for column C1. When writing column DECIMAL(8,4) into
database column DECIMAL(6,2), truncation, loss of precision or data
corruption can occur.
#2
Type: Warning
Message Id: IIS-CONN-DAAPI-00396
Message: DRS_Connector_1: Writing the WVARCHAR column C2 into
a VARCHAR database column can cause data loss or corruption due to
```

```
character set conversions.
#3
Type: Warning
Message Id: IIS-CONN-DAAPI-00393
Message: DRS_Connector_1: The length of WVARCHAR column C2
cannot be validated because the database column is VARCHAR and character
set conversion is involved. Inadequate column lengths can lead to
data truncation or unexpected errors.
#4
Type: Warning
Message Id: IIS-CONN-DAAPI-00398
Message: DRS_Connector_1: Schema reconciliation detected a
size mismatch for column C3. When writing column VARCHAR(min=0,max=10)
into database column VARCHAR(min=0,max=5), truncation, loss of precision
or data corruption can occur.
#5
Type: Warning
Message Id: IIS-CONN-DAAPI-00398
Message: DRS_Connector_1: Schema reconciliation detected a
size mismatch for column C4. When writing column SFLOAT into database
column INT32, truncation, loss of precision or data corruption can
occur.
#6
Type: Warning
Message Id: IIS-CONN-DAAPI-00398
Message: DRS_Connector_1: Schema reconciliation detected a
size mismatch for column C5. When writing column DATETIME(fraction=6)
into database column DATETIME(fraction=3), truncation, loss of precision
or data corruption can occur.
```

The job log can report additional warnings or fatal errors, depending on the actual data on the link and the type of the stage on the other side of the link.

Message #1 is reported because the Decimal(8,4) column C1 on the link has larger precision and scale than the corresponding DECIMAL(6,2) column C1 in the target table. To eliminate this warning, the Length and Scale attributes for the column C1 on the link should be set to values 6 and 2 (or smaller) so that they do not exceed the precision and scale of the column C1 in the target table.

Messages #2 and #3 are reported because the Unicode column C2 on the link is used to write to a VARCHAR column in the database. To eliminate the warnings the type of column C2 on the link should be changed from NVarChar to VarChar.

Message #4 is reported because the VarChar column C3 on the link has larger length than the corresponding VARCHAR column C3 in the target table. To eliminate this warning, the Length attribute for the column C3 on the link should be set to value 5 or smaller so that it doesn't exceed the length of the column C3 in the target table.

Message #5 is reported because the Float column C4 on the link is used to write to an INTEGER column C4 in the target table. The range of allowed values for the Float InfoSphere DataStage type is larger than the range of allowed values for the INTEGER DB2 type. To eliminate this warning, the type of the column C4 on the link should be changed from Float to Integer (or to SmallInt or TinyInt).

Message #6 is reported because the Timestamp column C5 on the link has larger fractional second precision (6) than the TIMESTAMP(3) column C5 in the target table. To eliminate this warning, the Extended attribute for the column C5 on the link should cleared. The Scale attribute should be cleared as well or set to the value 3 or smaller so that it doesn't exceed the fractional second precision of the column C5 in the target table.

An alternative approach for eliminating messages #1 to #6 would be to modify column definitions in the target table so that they are in agreement with the column definitions on the link.

Yet another approach would be to define message handler for the job and demote the messages with the corresponding message identifiers from Warning severity to Informational severity. For the example used here, the message identifiers that would need to be demoted are: IIS-CONN-DAAPI-00398 (for messages #1, #4, #5 and #6), IIS-CONN-DAAPI-00396 (for message #2) and IIS-CONN-DAAPI-00393 (for message #3). Note that demoting or suppressing schema reconciliation messages reported by the DRS Connector stage will have no effect on any related warning and error messages logged by the InfoSphere DataStage framework and other stages in the job.

Example 2

In this example a parallel InfoSphere DataStage job contains a DRS Connector stage configured for DB2 database type. The stage is used to read data from a DB2 database table. The output link of the stage contains the following columns:

Table 13. Example 2: Output link contains the following columns

Name	Type	Length	Scale	Nullable	Extended
C1	Decimal	6	2	No	(n/a)
C2	VarChar	10	(n/a)	No	(not set)
C3	VarChar	3	(n/a)	No	(not set)
C4	Float	(n/a)	(n/a)	No	(n/a)
C5	Date	(n/a)	(n/a)	No	(n/a)

The source table in the database contains the following columns:

Table 14. Example 2: Source table contains the following columns

Nsme	Type	Length	Scale	Nullable
C1	DECIMAL	8	4	No
C2	VARGRAPHIC	10	(n/a)	No
C3	VARCHAR	5	(n/a)	Yes
C4	FLOAT	(n/a)	(n/a)	No
C5	TIMESTAMP	(n/a)	3	No

The job runs and the following messages are reported in the job log:

```
#1
Type: Warning
Message Id: IIS-CONN-DAAPI-00399
Message: DRS_Connector_0: Schema reconciliation detected a size
mismatch for column C1. When reading database column DECIMAL(8,4)
into column DECIMAL(6,2), truncation, loss of precision or data corruption
can occur.
#2
Type: Warning
Message Id: IIS-CONN-DAAPI-00397
Message: DRS_Connector_0: Reading the WVARCHAR database column
C2 into a VARCHAR column can cause data loss or corruption due to
character set conversions.
#3
```

```
Type: Warning
Message Id: IIS-CONN-DAAPI-00393
Message: DRS_Connector_0: The length of VARCHAR column C2 cannot
be validated because the database column is WVARCHAR and character
set conversion is involved. Inadequate column lengths can lead to
data truncation or unexpected errors.
#4
Type: Warning
Message Id: IIS-CONN-DAAPI-00399
Message: DRS_Connector_0: Schema reconciliation detected a size
mismatch for column C3. When reading database column VARCHAR(min=0,max=5)
into column VARCHAR(min=0,max=3), truncation, loss of precision or
data corruption can occur.
#5
Type: Warning
Message Id: IIS-CONN-DAAPI-00399
Message: DRS_Connector_0: Schema reconciliation detected a size
mismatch for column C4. When reading database column DFLOAT into column
SFLOAT, truncation, loss of precision or data corruption can occur.
#6
Type: Warning
Message Id: IIS-CONN-DAAPI-00399
Message: DRS_Connector_0: Schema reconciliation detected a size
mismatch for column C5. When reading database column DATETIME(fraction=3)
into column DATE, truncation, loss of precision or data corruption
can occur.
#7
Type: Info
Message Id: IIS-CONN-DAAPI-00057
Message: DRS_Connector_0: Modified field: C5, attribute: IS_NULLABLE.
Design-time value: 0. External value: 1
```

The message #1 is reported because the DECIMAL(8,4) column C1 in the database has larger precision and scale than the corresponding Decimal(6,2) column C1 on the link. To eliminate this warning, the Length and Scale attributes for the column C1 on the link should be set to values 8 and 4 (or larger) so that they are not smaller than the precision and scale of the column C1 in the source table.

The messages #2 and #3 are reported because the VarChar column C2 on the link is used to read data from the VARGRAPHIC column C2 in the database. To eliminate the warnings the type of the column C2 on the link should be changed from VarChar to NVarChar (or Extended attribute for this column should be set to value Unicode).

The message #4 is reported because the VARCHAR(5) column C3 in the database has larger length than the corresponding VarChar(3) column C3 on the link. To eliminate this warning, the Length attribute for the column C3 on the link should be set to value 5 (or larger) so that it is not smaller than the length of the column C3 in the source table.

The message #5 is reported because the Float column C4 on the link is used to read data from the FLOAT column C4 in the target table. The range of allowed values for the FLOAT DB2 type is larger than the range of allowed values for the Float InfoSphere DataStage type. To eliminate this warning, the type of the column C4 on the link should be changed from Float to Double.

The message #6 is reported because the Date column C5 on the link does not support hours, minutes, seconds and fractional seconds and this column is used to read data from the TIMESTAMP(3) column C5 in the source table. To eliminate this warning, the type of the column C5 on the link should changed from Date to Timestamp. Also the Scale attribute should be set to value 3 or larger or the

Extended attribute should be set to Microseconds. That way the fractional second precision for the column C5 on the link will not be smaller than the fractional second precision for the column C5 in the source table.

The message #7 is of informational severity and is reported because the column C5 in the source table supports NULL values, and the column C5 on the link does not. To eliminate this message, it is necessary to change the Nullable attribute for column C5 on the link from No to Yes.

An alternative approach for eliminating messages #1 to #7 would be to modify column definitions in the source table so that they are in agreement with the column definitions on the link.

Yet another approach would be to define message handler for the job and demote the messages with the corresponding message identifiers from Warning severity to Informational severity. For the example used here, the message identifiers that would need to be demoted are: IIS-CONN-DAAPI-00399 (messages #1, #4, #5 and #6), IIS-CONN-DAAPI-00397 (message #2) and IIS-CONN-DAAPI-00393 (message #3). The message #7 with message identifier IIS-CONN-DAAPI-00057 is already at the Informational severity so the message handler cannot be configured to further demote it. However it can be configured to suppress this message from the log. Note that demoting or suppressing schema reconciliation messages reported by the DRS Connector stage will have no effect on any related warning and error messages logged by the InfoSphere DataStage framework and other stages in the job.

Column aliases

When the DRS Connector stage is configured for the IBM DB2 and ODBC database types, the items that are specified in a `SELECT` statement must correspond to the column names on the output link of the stage.

For example, for SQL expressions in the select list the use of column aliases is required since the actual text of the expressions will not match the column names on the link. This applies to the native SQL expressions specific to the database type for which the stage is configured as well as to the SQL meta tags supported by the DRS Connector stage.

When a Dynamic RDBMS stage is migrated to a DRS Connector stage that is configured for IBM DB2 or ODBC database type and the migrated stage specifies `SELECT` statement which does not use aliases for the SQL expressions in the statement, you must add column aliases to the expressions so that they correspond to the column names on the link.

The DRS Connector stage configured for the Oracle database type and the Dynamic RDBMS stage allow the use of `SELECT` statements where items in the list do not match column names on the output link. This includes the case when SQL expressions are used without column aliases. Note though that the use of column aliases is highly recommended in this case as well.

For example, if the DRS Connector stage is configured to read data from an IBM DB2 or ODBC database type, and the output link of the stage contains columns C1, C2 and C3, the following statement with SQL meta tags will not work:

```
SELECT %ABS(C1), %MIN(C3), %MAX(C2) FROM TABLE1
```

For this statement to work, aliases would need to be provided for the SQL expressions used in the statement and those aliases would need to correspond to the column names on the link with which the SQL expressions should be associated.

For example the following statement would associate the expression %ABS(C1) with the column C1 on the link, the expression %MIN(C3) with the column C3 on the link and the expression %MAX(C2) with the column C2 on the link:

```
SELECT %ABS(C1) C1, %MIN(C3) C3, %MAX(C2) C2 FROM TABLE1
```

For the DRS Connector stage configured for Oracle database type and for the Dynamic RDBMS stage the statement without aliases will work and the stage will perform association between the expressions and the columns on the link in an ordered fashion: the expression %ABS(C1) will be associated with the column C1 on the link, the expression %MIN(C3) will be associated with the column C2 on the link and the expression %MAX(C2) will be associated with column C3 on the link. In some cases such an association strategy can be intended, but in other cases it may not be. To prevent the stage from performing association in the ordered fashion it is highly recommended to explicitly include column aliases that correspond to the names of the columns on the link, so that the stage can perform the association by name.

Handling TO_DATE() and TO_CHAR() SQL functions for the Oracle database type

The Dynamic RDBMS stage uses TO_DATE and TO_CHAR Oracle SQL functions to fetch data and write data for InfoSphere DataStage Date, Time and Timestamp data types.

For example, when this stage is configured to auto-generate SELECT statement and column C1 on the output link is defined as Date column, column C2 as Time column and column C3 as Timestamp column, they will appear in the select list of the generated `SELECT` statement respectively as:

```
TO_CHAR(C1, 'YYYY-MM-DD')
TO_CHAR(C2, 'HH24:MI:SS')
TO_CHAR(C3, 'YYYY-MM-DD HH24:MI:SS')
```

If the stage is configured to auto-generate an INSERT statement and column C1 on the input link is defined as Date column, column C2 as Time column and column C3 as Timestamp column, they will appear in the parameter list of the generated INSERT statement respectively as:

```
TO_DATE(:1, 'YYYY-MM-DD')
TO_DATE(:2, 'HH24:MI:SS')
TO_DATE(:3, 'YYYY-MM-DD HH24:MI:SS')
```

The user-defined SQL statements in Dynamic RDBMS stage also use TO_CHAR and TO_DATE SQL functions for Date, Time and Timestamp columns.

The DRS Connector stage does not require these functions for Date, Time and Timestamp columns. It exchanges data with the database using internal Oracle date and timestamp datatypes so no conversion to and from character data is involved.

When a Dynamic plug-in stage with TO_CHAR and TO_DATE functions in user-defined SQL statements is migrated to the DRS Connector stage, the jobs can fail due to this difference. The problem does not occur for migrated stages with

auto-generated SQL statement, because the SQL statement that the DRS Connector stage will generate at runtime will not use TO_CHAR and TO_DATE functions like if those functions were used in the auto-generated statement generated at runtime by the Dynamic RDBMS stage in the original job. This is because auto-generated statements are always generated from scratch when the job runs and the mechanism used by the Dynamic RDBMS stage does not have effect on the mechanism used by the DRS Connector stage.

The preferred way for resolving job failures dues to use of TO_CHAR and TO_DATE SQL functions in the user-generated SQL statement of the Dynamic RDBMS stage after migrating it to DRS Connector stage is to remove those functions from the statement.

For example if the user-defined `SELECT` statement contains expression TO_CHAR(C1, 'YYYY-MM-DD') in the select list, replace it with C1.

As another example, if the user-defined INSERT statement contains expression TO_DATE(:2, 'HH24-MI-SS') in the parameter list, replace it with :2.

Another way for resolving the problem is to modify types of the columns on the link that correspond to these SQL functions. The types should be changed to character type (such as Char or VarChar) with the length that corresponds to the date/time format used in the SQL function.

For example if the user-defined `SELECT` statement contains expression TO_CHAR(C1, 'YYYY-MM-DD') in the select list, change the data type for column C1 on the link from Date to Char(10) since 10 is the total number of character positions for date values in 'YYYY-MM-DD' format (for example 2010-09-27).

As another example, if the user-defined INSERT statement contains expression TO_DATE(:2, 'HH24-MI-SS') in the parameter list, replace the column on the link that corresponds to first parameter from Time to Char(10) since 10 is the total number of character positions for time values in 'HH24-MI-SS' format (for example 11:26:45).

Finally in cases when modifying the existing statements or column definitions on the links is not convenient, another option is to define *CC_ORA_BIND_DATETIME_AS_CHAR* environment variable in the InfoSphere DataStage project and set it to value TRUE. Alternatively it can be defined at the project level with the default value of FALSE and added to individual jobs as job parameter and overridden to value TRUE for them.

When *CC_ORA_BIND_DATETIME_AS_CHAR* environment variable is defined and set to TRUE, the DRS Connector stage uses character representation for Date and Timestamp values exchanged with the Oracle database, using the same date and time formats used by the Dynamic RDBMS stage. That way the connector mimics the behavior of the Dynamic RDBMS stage. The use of this environment variable is not recommended in general case since it forces the behavior in the DRS Connector which is added only for backward compatibility with Dynamic RDBMS stage and which involves data conversions that are normally not necessary and which in turn can have significant negative impact on the performance.

Oracle manual load connection settings

The DRS Connector stage must have a valid connection to an Oracle database to bulk load data manually. When the Dynamic RDBMS stage bulk loads data

manually, it does not write data to the database table directly but instead produces a pair of control and data files. The control and data files can then be used by the SQL*Loader Oracle utility to load the data to the database.

When the Dynamic RDBMS stage performs manual load, it does not attempt to connect to the database. Consequently it is possible to provide invalid database connection information and invalid table name in the stage properties and the job will still complete successfully.

The DRS Connector stage also supports manual load mode for Oracle database type and like the Dynamic RDBMS stage it also creates a pair of control and data files. However, the DRS Connector will always attempt to connect to the database based on the connection property values specified for the stage. If the connection fails the job also fails.

When a job with the Dynamic RDBMS stage configured for Oracle manual load is migrated to use the DRS Connector stage, the job will fail if the database connection information in the stage is invalid. In order to be able to run the job successfully it is necessary to edit the connection properties in the stage and set them to valid Oracle connection values. The stage will then be able to connect to the database and the job will proceed.

Note: This requirement exists even if the connector does not use the connection to create control and data files. On the other hand, the table name specified in the stage does not have to match an existing table in the database.

Appendix A. Product accessibility

You can get information about the accessibility status of IBM products.

The IBM InfoSphere Information Server product modules and user interfaces are not fully accessible.

For information about the accessibility status of IBM products, see the IBM product accessibility information at http://www.ibm.com/able/product_accessibility/index.html.

Accessible documentation

Accessible documentation for InfoSphere Information Server products is provided in an information center. The information center presents the documentation in XHTML 1.0 format, which is viewable in most web browsers. Because the information center uses XHTML, you can set display preferences in your browser. This also allows you to use screen readers and other assistive technologies to access the documentation.

The documentation that is in the information center is also provided in PDF files, which are not fully accessible.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility.

Appendix B. Contacting IBM

You can contact IBM for customer support, software services, product information, and general information. You also can provide feedback to IBM about products and documentation.

The following table lists resources for customer support, software services, training, and product and solutions information.

Table 15. IBM resources

Resource	Description and location
IBM Support Portal	You can customize support information by choosing the products and the topics that interest you at www.ibm.com/support/entry/portal/Software/Information_Management/InfoSphere_Information_Server
Software services	You can find information about software, IT, and business consulting services, on the solutions site at www.ibm.com/businesssolutions/
My IBM	You can manage links to IBM Web sites and information that meet your specific technical support needs by creating an account on the My IBM site at www.ibm.com/account/
Training and certification	You can learn about technical training and education services designed for individuals, companies, and public organizations to acquire, maintain, and optimize their IT skills at http://www.ibm.com/training
IBM representatives	You can contact an IBM representative to learn about solutions at www.ibm.com/connect/ibm/us/en/

Appendix C. Accessing the product documentation

Documentation is provided in a variety of formats: in the online IBM Knowledge Center, in an optional locally installed information center, and as PDF books. You can access the online or locally installed help directly from the product client interfaces.

IBM Knowledge Center is the best place to find the most up-to-date information for InfoSphere Information Server. IBM Knowledge Center contains help for most of the product interfaces, as well as complete documentation for all the product modules in the suite. You can open IBM Knowledge Center from the installed product or from a web browser.

Accessing IBM Knowledge Center

There are various ways to access the online documentation:

- Click the **Help** link in the upper right of the client interface.
- Press the F1 key. The F1 key typically opens the topic that describes the current context of the client interface.

  Note: The F1 key does not work in web clients.
- Type the address in a web browser, for example, when you are not logged in to the product.

  Enter the following address to access all versions of InfoSphere Information Server documentation:

  `http://www.ibm.com/support/knowledgecenter/SSZJPZ/`

  If you want to access a particular topic, specify the version number with the product identifier, the documentation plug-in name, and the topic path in the URL. For example, the URL for the 11.3 version of this topic is as follows. (The ⇒ symbol indicates a line continuation):

  ```
  http://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/⇒
  com.ibm.swg.im.iis.common.doc/common/accessingiidoc.html
  ```

  Tip:

  The knowledge center has a short URL as well:

  `http://ibm.biz/knowctr`

  To specify a short URL to a specific product page, version, or topic, use a hash character (#) between the short URL and the product identifier. For example, the short URL to all the InfoSphere Information Server documentation is the following URL:

  `http://ibm.biz/knowctr#SSZJPZ/`

  And, the short URL to the topic above to create a slightly shorter URL is the following URL (The ⇒ symbol indicates a line continuation):

  ```
  http://ibm.biz/knowctr#SSZJPZ_11.3.0/com.ibm.swg.im.iis.common.doc/⇒
  common/accessingiidoc.html
  ```

Changing help links to refer to locally installed documentation

IBM Knowledge Center contains the most up-to-date version of the documentation. However, you can install a local version of the documentation as an information center and configure your help links to point to it. A local information center is useful if your enterprise does not provide access to the internet.

Use the installation instructions that come with the information center installation package to install it on the computer of your choice. After you install and start the information center, you can use the **iisAdmin** command on the services tier computer to change the documentation location that the product F1 and help links refer to. (The ⇒ symbol indicates a line continuation):

Windows

```
IS_install_path\ASBServer\bin\iisAdmin.bat -set -key ⇒
com.ibm.iis.infocenter.url -value http://<host>:<port>/help/topic/
```

AIX Linux

```
IS_install_path/ASBServer/bin/iisAdmin.sh -set -key ⇒
com.ibm.iis.infocenter.url -value http://<host>:<port>/help/topic/
```

Where <host> is the name of the computer where the information center is installed and <port> is the port number for the information center. The default port number is 8888. For example, on a computer named server1.example.com that uses the default port, the URL value would be http://server1.example.com:8888/help/topic/.

Obtaining PDF and hardcopy documentation

- The PDF file books are available online and can be accessed from this support document: https://www.ibm.com/support/docview.wss?uid=swg27008803&wv=1.
- You can also order IBM publications in hardcopy format online or through your local IBM representative. To order publications online, go to the IBM Publications Center at http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss.

Appendix D. Providing feedback on the product documentation

You can provide helpful feedback regarding IBM documentation.

Your feedback helps IBM to provide quality information. You can use any of the following methods to provide comments:

- To provide a comment about a topic in IBM Knowledge Center that is hosted on the IBM website, sign in and add a comment by clicking **Add Comment** button at the bottom of the topic. Comments submitted this way are viewable by the public.
- To send a comment about the topic in IBM Knowledge Center to IBM that is not viewable by anyone else, sign in and click the **Feedback** link at the bottom of IBM Knowledge Center.
- Send your comments by using the online readers' comment form at www.ibm.com/software/awdtools/rcf/.
- Send your comments by e-mail to comments@us.ibm.com. Include the name of the product, the version number of the product, and the name and part number of the information (if applicable). If you are commenting on specific text, include the location of the text (for example, a title, a table number, or a page number).

Notices and trademarks

This information was developed for products and services offered in the U.S.A. This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

Notices

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session or persistent cookies. If a product or component is not listed, that product or component does not use cookies.

Table 16. Use of cookies by InfoSphere Information Server products and components

Product module	Component or feature	Type of cookie that is used	Collect this data	Purpose of data	Disabling the cookies
Any (part of InfoSphere Information Server installation)	InfoSphere Information Server web console	• Session • Persistent	User name	• Session management • Authentication	Cannot be disabled
Any (part of InfoSphere Information Server installation)	InfoSphere Metadata Asset Manager	• Session • Persistent	No personally identifiable information	• Session management • Authentication • Enhanced user usability • Single sign-on configuration	Cannot be disabled

Table 16. Use of cookies by InfoSphere Information Server products and components (continued)

Product module	Component or feature	Type of cookie that is used	Collect this data	Purpose of data	Disabling the cookies
InfoSphere DataStage	Big Data File stage	• Session • Persistent	• User name • Digital signature • Session ID	• Session management • Authentication • Single sign-on configuration	Cannot be disabled
InfoSphere DataStage	XML stage	Session	Internal identifiers	• Session management • Authentication	Cannot be disabled
InfoSphere DataStage	IBM InfoSphere DataStage and QualityStage Operations Console	Session	No personally identifiable information	• Session management • Authentication	Cannot be disabled
InfoSphere Data Click	InfoSphere Information Server web console	• Session • Persistent	User name	• Session management • Authentication	Cannot be disabled
InfoSphere Data Quality Console		Session	No personally identifiable information	• Session management • Authentication • Single sign-on configuration	Cannot be disabled
InfoSphere QualityStage Standardization Rules Designer	InfoSphere Information Server web console	• Session • Persistent	User name	• Session management • Authentication	Cannot be disabled
InfoSphere Information Governance Catalog		• Session • Persistent	• User name • Internal identifiers • State of the tree	• Session management • Authentication • Single sign-on configuration	Cannot be disabled
InfoSphere Information Analyzer	Data Rules stage in the InfoSphere DataStage and QualityStage Designer client	Session	Session ID	Session management	Cannot be disabled

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/software/info/product-privacy.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Intel and Itanium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The United States Postal Service owns the following trademarks: CASS, CASS Certified, DPV, LACS^{Link}, ZIP, ZIP + 4, ZIP Code, Post Office, Postal Service, USPS and United States Postal Service. IBM Corporation is a non-exclusive DPV and LACS^{Link} licensee of the United States Postal Service.

Other company, product or service names may be trademarks or service marks of others.

Index

Printed in USA

SC19-4271-00